Escuela Universitaria Ingeniería Técnica Industrial ZARAGOZA

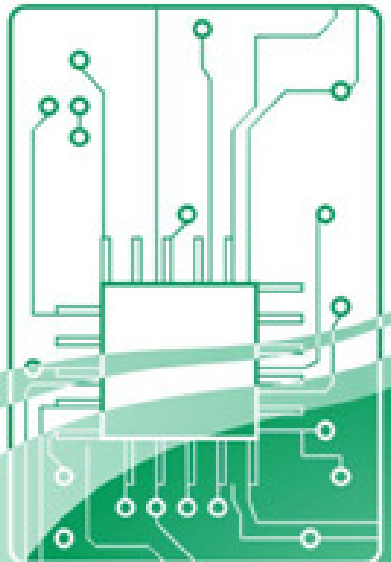
Universidad Zaragoza 1542

Efectos musicales digitales. Programación alternativa mediante el uso de la Transformada Wavelet Continua Compleja (CCWT)

Proyecto Fin de Carrera

Memoria

Alumno: **Óscar Gil Bailo**
Especialidad: **Electrónica Industrial**
Director: **Jesús Ponce de León Vázquez**
Convocatoria: **Septiembre 2011**



Efectos musicales digitales. Programación alternativa mediante el uso de la Transformada Wavelet Continua Compleja (CCWT)

RESUMEN

El presente proyecto fin de carrera ha consistido en la programación de un total de 19 efectos musicales utilizando, para ello, un método de análisis y síntesis de señales de audio desarrollado en la Universidad de Zaragoza, denominado Algoritmo de Síntesis Aditiva por Wavelets Complejas (CWAS por sus siglas en inglés). Para la programación de dichos efectos se ha utilizado el lenguaje y las herramientas de programación que nos brinda la aplicación informática *Matlab*.

La mayoría de los efectos analizados en este proyecto se encuentran recogidos en el libro *Digital Audio Effects (DAFX)*, considerado como una importante referencia en el campo de los efectos de audio digitales. En este libro se describen los métodos tradicionales utilizados hasta el momento para la implementación de distintos efectos, basados en su mayoría en el uso de la transformada corta de Fourier (STFT). Además, en este libro se muestran también los algoritmos de programación desarrollados en *Matlab* para algunos de los efectos.

Por lo tanto, además de desarrollar los *scripts* de *Matlab* para la implementación de distintos efectos musicales a partir del algoritmo CWAS, los resultados obtenidos para cada efecto (forma de onda, escalograma, resultado acústico…) se han comparado con los ofrecidos por los correspondientes métodos tradicionales (en aquellos casos en los que se está en posesión del *script* tradicional). Por otro lado, cabe destacar que algunos de los efectos implementados en este proyecto no se encuentran recogidos en el libro DAFX, es decir, se ha introducido cierto grado de innovación en el desarrollo o en la evolución de algunos de los efectos analizados.

Puesto que en el campo del audio no existen parámetros estandarizados que permitan evaluar la calidad final de los efectos musicales, la mejorar manera de valorar los resultados obtenidos en este proyecto es escuchando los resultados sonoros. Para ello se han generado diversos archivos de audio de cada efecto analizado en los que se puede escuchar el sonido original, el sonido resultante del efecto basado en el algoritmo CWAS y el sonido resultante del efecto tradicional (en aquellos casos en los que se está en posesión del *script* tradicional).

De forma general, tanto los resultados gráficos como los resultados acústicos de la mayor parte de los efectos desarrollados a partir del algoritmo CWAS han mostrado una calidad superior a la que se consigue con los métodos tradicionales, sin necesidad de recurrir a algoritmos de programación excesivamente complejos. Por tanto, el método CWAS puede suponer una evolución para algunos efectos tradicionales, consiguiendo efectos mucho más complejos y musicales.

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS

1.1. OBJETIVO

El presente proyecto fin de carrera tiene como objetivo la programación alternativa de efectos musicales utilizando el lenguaje y las herramientas de programación que nos brinda la aplicación informática *Matlab* [1].

Para la realización de este proyecto se ha utilizado un método de análisis y síntesis de señales de audio desarrollado en la Universidad de Zaragoza, denominado algoritmo de Síntesis Aditiva por Wavelets Complejas (CWAS por sus siglas en inglés) [2]. A diferencia de los métodos de análisis tradicionales, basados en el uso de la transformada corta de Fourier (STFT), dicho método se basa en el uso de la transformada Wavelet.

Un total de 19 efectos musicales han sido programados a partir del algoritmo CWAS (dichos algoritmos de programación se han incluido en el documento *Anexos*). Los algoritmos de programación a partir de métodos tradicionales de muchos de estos efectos musicales se describen detalladamente en el libro *Digital Audio Effects (DAFx)* [3]. Con el objetivo de evaluar las posibles mejoras que puede introducir la aplicación del algoritmo CWAS en la implementación de efectos musicales, se ha realizado una comparación entre los resultados ofrecidos por ambos métodos (algoritmo CWAS y método tradicional). Varios de los efectos que aparecen a lo largo del proyecto se pueden implementar con distintos métodos tradicionales. Sin embargo, para la comparación de ambos métodos, se ha elegido un único método tradicional para cada efecto, buscando siempre la mejor relación entre sencillez de programación y calidad final del sonido.

El objetivo del proyecto es, por tanto, aprovechar las ventajas que brinda el uso de la herramienta CWAS para mejorar los resultados de los efectos musicales planteados. Además, algunos de los efectos más complejos que han sido programados a partir del algoritmo CWAS en este proyecto no habían sido realizados previamente mediante métodos tradicionales (no aparecen en *DAFx*), por lo que puede decirse que incluyen cierto grado de innovación.

Puesto que no existen parámetros estandarizados que permitan evaluar la calidad final de los efectos musicales, la mejor forma de evaluarla es escuchar los resultados acústicos. Para ello se han generado diversos archivos de audio de cada efecto analizado. Debido a que en el repositorio de documentos electrónicos de la Universidad de Zaragoza (*Zaguán*) sólo pueden cargarse archivos en formato *pdf*, los archivos de audio generados se encuentran a disposición del lector en el siguiente enlace: http://www.megaupload.com/?d=UTR6I1ZP

1.2. LOS EFECTOS MUSICALES

Los efectos musicales son dispositivos físicos o herramientas software que modifican el sonido de un instrumento o de cualquier otra fuente de sonido mediante una serie de parámetros de control. Estos efectos pueden aplicarse en tiempo real, lo que significa que el músico puede utilizarlos en plena actuación (lo más típico es en guitarras eléctricas, bajos eléctricos, teclados, voz,…) o en la fase de post-producción sobre los canales de audio ya grabados.

Los efectos musicales son utilizados por todos los individuos que participan en la creación de música, desde el músico que se sirve de ellos para la creación de técnicas de ejecución especiales, hasta los procesadores de efectos usados para sintetizar, grabar o producir señales musicales.

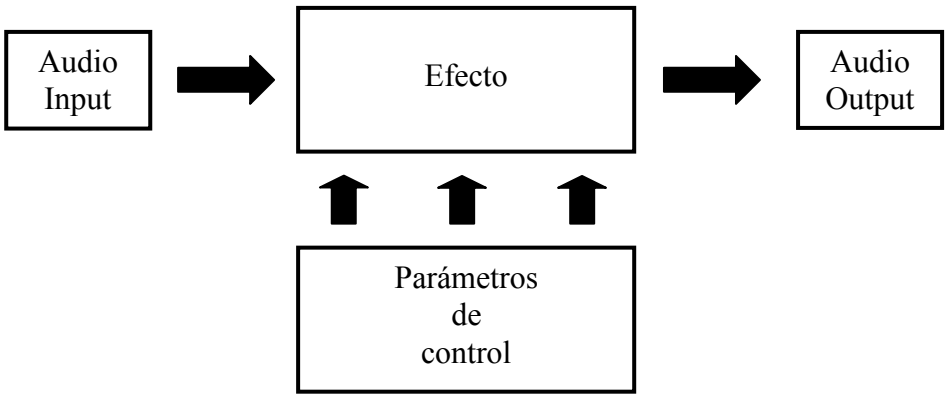


Figura 1.1. Uso de los efectos musicales.

Los efectos musicales se pueden clasificar en dos categorías:

- Efectos en formato físico.

Los efectos en formato físico se suelen aplicar en tiempo real y, generalmente, su uso se asocia con instrumentos como la guitarra eléctrica, el bajo eléctrico o el teclado. Normalmente se comercializan en formato pedal (stomp box) o bien en formato rack.

Figura 1.2. Efectos en formato pedal.

Figura 1.3. Efectos en *rack*.

A su vez, los efectos en formato físico se pueden clasificar en las siguientes categorías:

- Efectos analógicos: Están compuestos por circuitos electrónicos analógicos (*overdrives*, ecualizadores, distorsiones, *boosters*,…) o por elementos electro-mecánicos (reverberación de muelles, *tape-echo*,…). Generalmente, las unidades de efectos analógicos suelen contener uno o dos tipos de efectos en el mismo paquete.

- Efectos digitales: Están compuestos por circuitos digitales. En ellos se realizan las siguientes etapas:

    - Conversión Analógica/Digital (A/D).
    - Procesamiento de la señal.
    - Conversión Digital/Analógica (D/A).

  Se pueden encontrar unidades de efectos digitales que contengan uno o dos efectos distintos. Sin embargo, gracias a las ventajas que brinda la tecnología

digital, cada vez es más común que este tipo de efectos se comercialicen en formato multi-efectos y contengan una gran variedad de efectos distintos.

Figura 1.4. Multi-efectos digital.

- Efectos en formato software

El avance de la informática en los últimos años ha hecho que cada vez sea más común el uso de los ordenadores con propósitos de grabación o edición de sonidos. Ya no es necesario acudir a un estudio de grabación y desembolsar una gran cantidad de dinero para grabar, basta con disponer de un PC común, una tarjeta de audio capaz de transformar una señal analógica en una señal digital y herramientas de software de edición de sonidos (*Cubase, Adobe Audition, Pro Tools, Audacity*,…), que suelen llevar incorporado un paquete de efectos básicos.

El ordenador, ayudado por las herramientas software de las que disponga, trabajará con la señal digital y la grabará y retocará según las necesidades del usuario. Además, junto con estas herramientas de edición de sonidos, se pueden usar instrumentos virtuales (VSTi o RTAS), que son efectos musicales en formato software que permiten modificar los sonidos según convenga.

2. MÉTODOS DE ANÁLISIS DEL SONIDO

2.1. LOS EFECTOS DE AUDIO DIGITALES

El presente proyecto se centra en los efectos de audio digitales (DAFX) y, particularmente, en los DAFX en formato software. Se ha trabajado con señales que ya se encuentran en formato digital, por lo que basta con una pequeña introducción al tema para familiarizarse con el proceso general que se lleva a cabo.

En los efectos digitales, tanto la entrada como la salida están en formato digital, aunque representan señales de audio analógicas. El objetivo de los efectos digitales es la modificación de las características sonoras de la señal de entrada según una serie de parámetros de control, que pueden ser dados por el ingeniero de sonido, el músico, o simplemente por el propio efecto.

Antes de la etapa de procesado de las señales digitales es necesario transformar la señal de audio analógica en una señal digital. Para ello se utiliza un conversor analógico/digital (ADC). Una vez que se tiene la señal digital, ésta se pasa por el DAFX, donde se modificará el sonido según se necesite. Después, se reconstruye la señal analógica pasando la señal digital por un conversor digital/analógico (DAC) [4].

Una señal de audio natural está compuesta por una suma de distintas ondas sinusoidales que oscilan a distintas frecuencias. La frecuencia de vibración más grave es la que determina normalmente la frecuencia fundamental de los sonidos. Las restantes frecuencias, que suelen ser múltiplos de la frecuencia fundamental, se denominan armónicos o parciales. Cada tipo de instrumento tiene, por su construcción, una serie diferente de armónicos, que son los que definen su timbre y hacen que una misma nota tocada por dos instrumentos distintos no suene de la misma manera.

Queda de manifiesto que, a la hora de implementar cierto tipo de efectos digitales, el poder pasar de una representación en el dominio temporal a una representación en el dominio frecuencial es de gran utilidad. El objetivo de esto es descomponer el sonido en cada uno de sus armónicos y conocer la amplitud y frecuencia de éstos para poder modificarlos según se necesite.

2.2. ANÁLISIS TRADICIONAL. LA TRANSFORMADA LOCALIZADA DE FOURIER

Como se ha comentado anteriormente, para implementar cierto tipo de efectos digitales es necesario conocer la amplitud y frecuencia de los parciales que componen un sonido. Para ello se ha utilizado la transformada de Fourier de forma tradicional (ecuación 2.1) [5].

$$F[f(t)] = F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-j\omega t} \partial t \qquad \text{(ec. 2.1)}$$

La transformada de Fourier es la herramienta matemática que permite pasar del dominio temporal al dominio frecuencial. De manera general, la transformada de Fourier es un procedimiento matemático que mapea cualquier señal analógica estacionaria a una serie infinita de sinusoides, cada una de ellas con una determinada amplitud y fase.

Para adaptar el análisis de Fourier al dominio de las señales muestreadas, de duración finita y variable respecto al tiempo, se define la transformada localizada de Fourier (STFT).

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-j\omega t} \, dt \qquad \text{(ec. 2.2)}$$

En este caso, se divide la señal en pequeñas porciones o segmentos, denominados tramas de análisis. El contenido frecuencial o espectro se calcula en cada una de las tramas utilizando la transformada rápida de Fourier (FFT).

2.2.1. ENVENTANADO DE LA SEÑAL

La STFT impone una secuencia de ventanas temporales de la señal de entrada, es decir, divide la señal en fragmentos cortos (*short time*) delimitados en el tiempo por una función ventana. Una ventana es un tipo específico de envolvente que se aplica para un análisis espectral. La duración de la ventana está comprendida normalmente entre 1 ms y 1 s, y los segmentos suelen solaparse. A través del análisis espectral individual de cada segmento enventanado se obtiene una secuencia de medidas (de espectros), que constituyen el espectro variable a lo largo del tiempo o espectrograma.

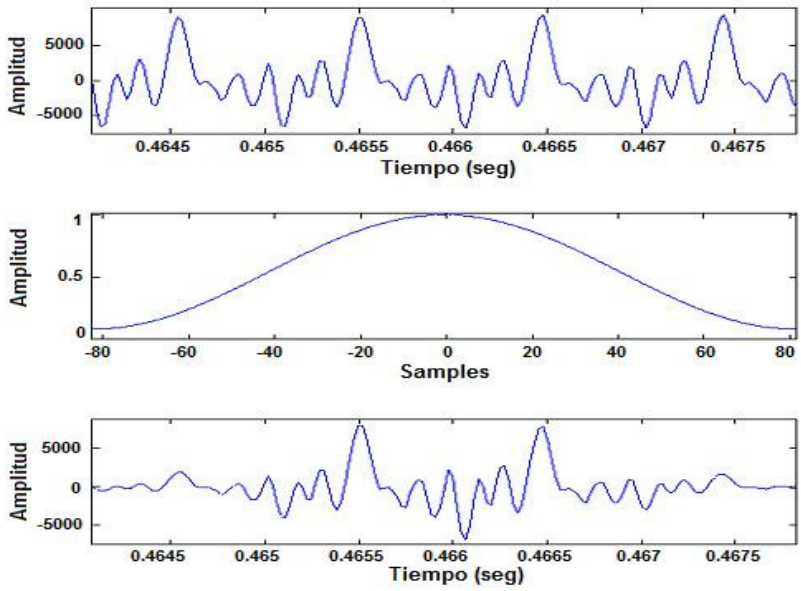


Figura 2.1. Proceso de enventanado de un segmento de audio.

Desafortunadamente el enventanado tiene la desventaja de producir distorsiones en la medida del espectro, ya que el analizador de espectro no mide sólo la señal de entrada, sino el producto de la misma por la ventana. El espectro que resulta es la convolución del espectro de la señal de entrada y el espectro de la ventana.

2.2.2. STFT

Tras el enventanado, la STFT aplica la transformada de Fourier discreta (DFT) (ecuación 2.3) a cada segmento enventanado. La DFT es un tipo de transformada de Fourier que puede trabajar con señales discretas en el tiempo, es decir, muestreadas (la transformada de Fourier rápida o FFT es una implementación eficiente de la DFT).

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) \cdot e^{-j 2\pi nk / N} \quad k = 0, 1, \ldots, N-1 \qquad \text{(ec. 2.3)}$$

Un problema de la FFT es el compromiso que debe establecerse entre resolución frecuencial y temporal. Si se quiere medir el contenido frecuencial de una señal de la forma más exacta posible, se necesitan muestras más grandes para analizar en cada tramo o segmento de la FFT. Pero si se hacen los segmentos muy grandes, no se capturan adecuadamente los eventos temporales que ocurren dentro de ellos. En otras palabras, más muestras requieren más tiempo, y cuanto más tiempo se considere el sonido, éste se parecerá menos a una sinusoide o a algo periódico, por lo que no estará bien representado por la FFT. En este caso, la STFT permite realizar un seguimiento de la fase de la señal ventana a ventana. Debido a la pérdida de resolución temporal del análisis, la información de fase no está disponible para todas las muestras de la señal y resulta necesario interpolar los valores de fase.

Por tanto, queda de manifiesto que, utilizando el método tradicional de análisis de sonidos mediante la STFT no se consigue un análisis suficientemente preciso de la señal para ciertas aplicaciones. Al reconstruir una señal sintética a partir de la STFT la señal puede tener un error bastante significativo.

2.3. LA TRANSFORMADA WAVELET CONTINUA COMPLEJA (CCWT)

En este apartado se resumen los aspectos más importantes de la Transformada Wavelet Continua (CWT).

La CWT se define mediante la ecuación 2.4.

$$W_f(a,b) = \int_{-\infty}^{+\infty} f(t) \cdot \psi_{a,b}^*(t) \cdot dt = f \cdot \overline{\psi}_a(b) \qquad \text{(ec. 2.4)}$$

donde * indica la conjugación compleja y $\psi_{a,b}(t)$ es la Wavelet madre (en este caso la Wavelet de Morlet) escalada un factor *a* y dilatada un factor *b* (ecuación 2.5).

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \qquad \text{(ec. 2.5)}$$

Desde el punto de vista del análisis de señal, la ecuación 2.4 representa el filtrado de la señal *f* con un filtro cuya respuesta al impulso es $\overline{\psi}_a(t)$.

$$\overline{\psi}_a(t) = \frac{1}{\sqrt{a}} \psi^*\left(\frac{-t}{a}\right) \qquad \text{(ec .2.6)}$$

La respuesta en frecuencia del filtro $\overline{\psi}_a(t)$ viene determinada por la ecuación 2.7.

$$\overline{\psi}_a(\omega) = \sqrt{a} \cdot \overline{\psi}(a\omega) \qquad \text{(ec. 2.7)}$$

La transformada Wavelet puede ser interpretada como el filtrado de una señal con un banco de filtros cuya respuesta en frecuencia es $\overline{\psi}_a(\omega)$.

En este trabajo la Wavelet utilizada es compleja, por tanto los coeficientes de la transformada son complejos y se puede extraer la parte real y la parte imaginaria de dichos coeficientes. A partir de las partes real e imaginaria de cada coeficiente es posible obtener la amplitud y la fase instantáneas de la señal. Un estudio detallado de esta relación se puede encontrar en [2].

2.4. *ANALISIS MEDIANTE LA TRANSFORMADA WAVELET. EL ALGORITMO CWAS*

En este apartado de la memoria se realiza una breve explicación del algoritmo para análisis de señales de audio CWAS o Síntesis Aditiva por Wavelets Complejas desarrollado por la Universidad de Zaragoza [2].

En la Figura 2.2 se muestra de forma gráfica el funcionamiento del algoritmo.

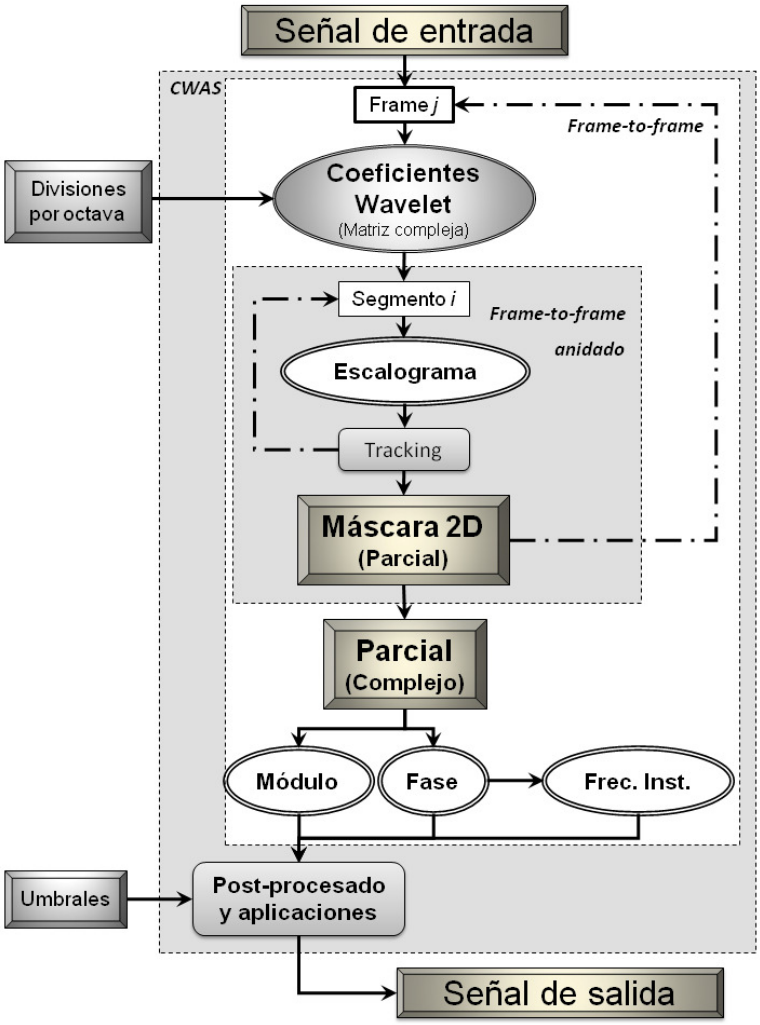


Figura 2.2. Esquema de funcionamiento del algoritmo CWAS.

El primer paso del algoritmo CWAS es el cálculo de la matriz de coeficientes Wavelet complejos partiendo de la Wavelet de Morlet (en adelante matriz CWT). Un estudio completo sobre cómo obtener estos coeficientes se puede encontrar en [2].

El tamaño de la matriz CWT será B*x*M, donde B es el número de bandas de frecuencia para el análisis (189 en este proyecto) y M el número de muestras o *samples* temporales de duración de la señal.

La matriz CWT puede ser estudiada en módulo y fase. En el módulo de los coeficientes se encuentra codificada toda la información para caracterizar la señal de audio original *x*(*t*). Este módulo se denomina Espectograma Wavelet y consiste en una representación tridimensional de la información contenida en los coeficientes complejos. A través del Escalograma Wavelet se realiza el *tracking* de parciales, mediante el cual se obtienen los parciales que forman el sonido y se almacenan en la matriz Parcial.

La matriz Parcial (formada por números complejos) tiene un tamaño N*x*M, donde N es el número de parciales del sonido analizado y M el número de *samples*.

A partir de la matriz Parcial se obtiene la matriz Módulo, que contiene los módulos instantáneos de cada parcial. La matriz Módulo se obtiene mediante el comando de *Matlab* indicado en la ecuación 2.8.

$$Módulo = ABS(Parcial) \qquad \text{(ec. 2.8)}$$

Del mismo modo se puede obtener la matriz Fase, que contiene las fases instantáneas de cada parcial, mediante el comando de *Matlab* indicado en la ecuación 2.9.

$$Fase = UNWRAP(ANGLE(Parcial)) \qquad \text{(ec. 2.9)}$$

Una vez que se han calculado los N parciales de la señal original, se reconstruye la señal sintética mediante un simple proceso de síntesis aditiva (ecuación 2.10).

$$x_{syn}(t) = \sum_{n=1}^{N} A_n(t)\cos[\phi_n(t)] \qquad \text{(ec. 2.10)}$$

La señal sintética conserva las cualidades tímbricas, frecuenciales, energéticas y de duración de la señal original.

El algoritmo CWAS permite obtener unos resultados de muy alta calidad en la resíntesis de señales, de manera que se obtiene una señal de error entre la señal original y la señal sintética muy pequeña (prácticamente despreciable). Dicho de otra forma, la señal original y la señal sintética resultan prácticamente indistinguibles tanto numérica como acústicamente.

Con el algoritmo CWAS se calculan de forma precisa los parciales que forman una señal de audio (con sus módulos, fases y frecuencias instantáneas), discernibles por el banco de filtros complejo empleado, por lo que resulta muy útil a la hora de implementar efectos musicales.

3. LISTADO DE EFECTOS

En este capítulo de la memoria se enumeran todos los efectos musicales con los que se ha trabajado a lo largo del proyecto, incluyendo también una breve descripción de su funcionamiento y programación mediante métodos tradicionales, así como de su implementación mediante el uso de los métodos basados en CWAS.

3.1. EFECTOS BASADOS EN RETARDOS (DELAY)

Una onda sonara que se refleja sobre una pared se suma a la onda sonora producida por la fuente, de manera que la onda que escucha el receptor es la suma de ambas. Si la pared en la que se refleja la onda está lo suficientemente lejos, el receptor escuchará la onda reflejada con un cierto retraso o *delay*. Este fenómeno es lo que se conoce como eco y constituye la base de una serie de efectos.

3.1.1. PROGRAMACIÓN TRADICIONAL

Para tratar de imitar este efecto, en los DAFX se usan una serie de estructuras básicas de *delay*. La más común es la que recibe el nombre de *FIR Comb Filter* (filtro peine de respuesta infinita). El efecto de esta estructura se escucha cuando la señal con *delay* se suma a la señal original, tomada como referencia.

Los parámetros que intervienen en un *FIR Comb Filter* son, por un lado, el tiempo de retardo o *delay* y, por otro lado, la ganancia "*g*" por la que se multiplica la señal retrasada.

$$y(n) = x(n) + g \times x(n - delay) \qquad \text{(ec. 3.1)}$$

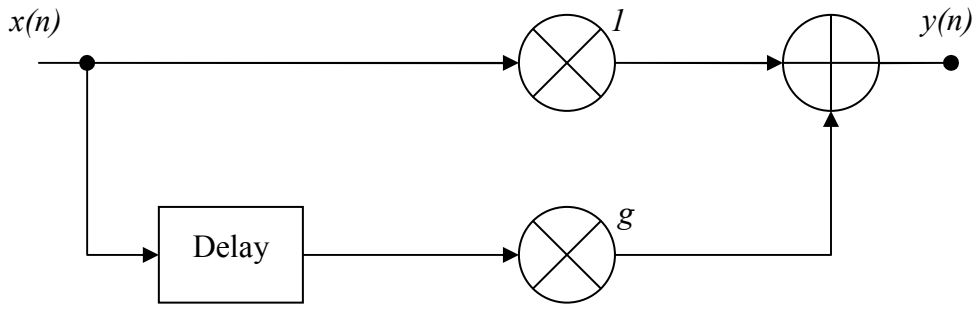


Figura 3.1. Estructura *FIR Comb Filter*.

Los efectos que se detallan a continuación están basados en esta sencilla estructura de *delay*.

3.1.1.1. Slapback

Este efecto se programa introduciendo en la estructura *FIR Comb Filter* un *delay* que oscila entre 10 y 25 ms. Consiste en una repetición muy rápida e inapreciable por el oído humano, pero que supone una alteración de la ecualización del sonido.

3.1.1.2. Echo (eco)

El efecto *Echo* simula lo que tradicionalmente se conoce como eco y consiste en la aplicación de un *delay* lo suficientemente grande como para que se escuche separación entre el sonido de la fuente y el sonido reflejado. Para programar este efecto se hace uso de la estructura *FIR Comb Filter*, introduciendo un *delay* mayor de 50 ms.

3.1.1.3. Chorus

Un *Chorus* (o coro en castellano) es un efecto que produce la sensación sonora de que dos o más personas están cantando o hablando a la vez. Cuando se aplica sobre un instrumento, produce la sensación de que varios instrumentos están ejecutando las mismas notas a la vez. Para programar el efecto *Chorus* se realizan tantas copias del sonido original como se desee, cada una de ellas con un *delay* que oscila de forma aleatoria entre 10 y 25 ms.

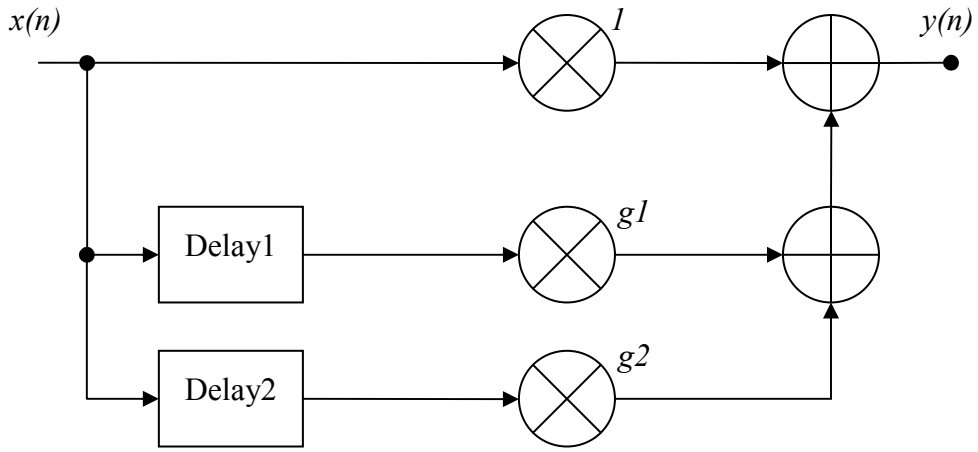


Figura 3.2. Estructura del efecto *Chorus*.

3.1.1.4. Vibrato

El *Vibrato* es una variación periódica de la frecuencia de un sonido. El efecto auditivo que se produce al aplicar *Vibrato* a una nota musical es que la nota base oscila y se mueve entre distintos tonos. Ejemplos de *Vibrato* se escuchan continuamente en los instrumentos de cuerda como violines o guitarras y también es fácil encontrarlos en la voz cantada.

Para programar un *Vibrato* por métodos tradicionales hay que tener en cuenta que, para esta aplicación, variar la distancia es lo mismo que variar el tiempo de *delay* o retraso. Si se continúa variando periódicamente el tiempo de *delay*, se consigue una variación periódica del tono, es decir, un efecto de *Vibrato* preciso. Para lograr el *Vibrato* se utiliza una línea de *delay* modulada por una señal sinusoidal de baja frecuencia (entre 5 y 14 Hz), de manera que siempre se tenga un tiempo de *delay* que oscile entre 5 y 10 ms.

En la Tabla 3.1 se recogen, a modo de resumen, los principales aspectos de los efectos basados en retardos que acaban de comentarse.

Tabla 3.1. Efectos basados en estructuras delay.

Rango de delay (ms)	***Tipo de modulación***	***Efecto***
5-10	Sinusoidal	Vibrato
10-25	Aleatoria	Chorus
>50	No	Echo
10-25	No	Slapback

3.1.2. PROGRAMACIÓN CWAS

En los efectos desarrollados a partir del algoritmo CWAS se trabaja con datos de sonido en forma de matrices (matriz Módulo y matriz Fase) de tamaño M*x*N, donde M será el tamaño en muestras de la señal de audio original y N será el número de parciales que tenga dicha señal. Lógicamente, si se compara el uso de matrices frente al uso de vectores (como hace la estructura *FIR Comb Filter*), el hecho de trabajar con matrices ralentiza el cálculo de la nueva señal punto por punto. Por este motivo se ha decidido implementar una nueva estructura *delay* que, aunque está basada en la estructura *FIR Comb Filter* y a efectos prácticos consigue el mismo resultado, reduce sensiblemente el tiempo de cálculo del algoritmo.

La nueva estructura de *delay* implementada se ha denominado estructura *Delay Zero-Padding* (en adelante DZP). La estructura DZP trabaja en bloque y consiste en aplicar un *Zero padding* al final de la matriz original, es decir, añadir D*x*N ceros al final de la misma, donde D es el número de *samples* de *delay* que va a tener la señal del efecto que se desea aplicar. De esta manera, la señal pasa a tener una duración equivalente a la duración de la señal original más la duración del tiempo de *delay* que se quiere aplicar.

En la Figura 3.3 se puede apreciar cómo se adapta la matriz de la señal original.

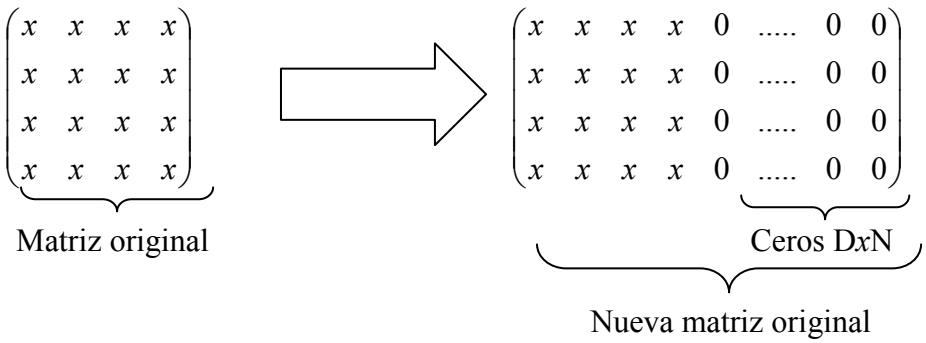


Figura 3.3. *Zero padding* de la señal original.

El siguiente paso que se realiza en la estructura DZP es duplicar la matriz original y aplicarle un *Zero padding* al inicio, de manera que se obtiene una matriz con el *delay* en *samples* que se desea (Figura 3.4).

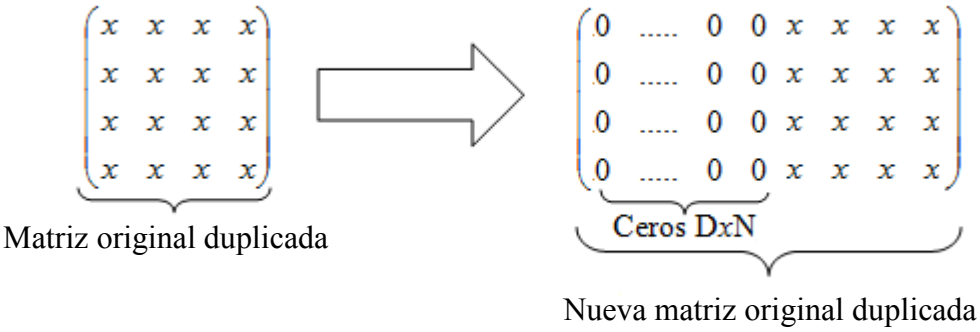


Figura 3.4. *Zero padding* de la señal duplicada.

3.1.2.1. Slapback-Echo

Como se ha comentado en apartados anteriores de la memoria, los efectos *Slapback* y *Echo* son dos efectos musicales basados en el uso de la estructura *FIR Comb Filter*. Ambos efectos se implementan de la misma manera variando únicamente el rango de *delay* de cada uno (ver Tabla 3.1).

Debido a la gran similitud entre ambos efectos, para implementarlos a partir del algoritmo CWAS se han considerado como un solo efecto, dando como resultado el efecto *Slapback-Echo*. En la Figura 3.5 se muestra un diagrama de bloques en el que se explica cómo se ha implementado el *Slapback-Echo*.

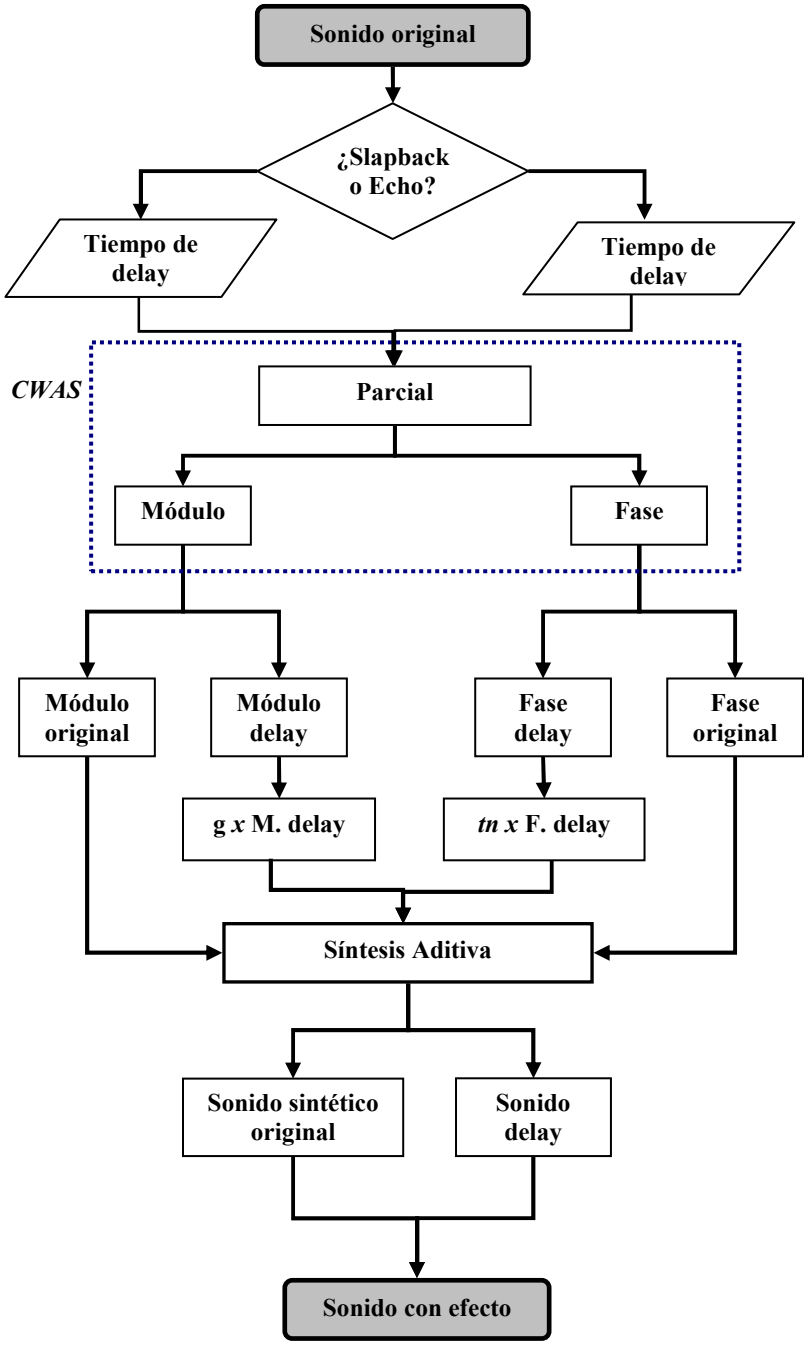


Figura 3.5. Diagrama del efecto *Slapback-Echo*.

Al inicio del algoritmo del efecto *Slapback-Echo*, el usuario selecciona el tipo de efecto que quiere aplicar a la señal original e indica el tiempo de *delay* deseado (en ms). A continuación, el tiempo de *delay* se transforma en muestras mediante el uso de la ecuación 3.2.

$$samples_{delay} = tiempo_{delay} \times \frac{Fs}{1000} \qquad \text{(ec. 3.2)}$$

donde el tiempo de *delay* se expresa en ms y *Fs* es el valor de la frecuencia de muestreo de la señal de audio original.

Sobre el sonido original se aplica el algoritmo CWAS y se obtiene la matriz Parcial, que contiene los datos de todos los parciales que componen el sonido. A partir de esta matriz se calculan las matrices Módulo y Fase que forman el sonido original. A estas matrices se les aplica la estructura DZP para adaptarlas a los *samples* de *delay* calculados en la ecuación 3.2 y se generan las matrices Módulo *delay* y Fase *delay*, que serán una copia de Módulo y Fase con un *zero-padding* al comienzo de la matriz. Esto hace que se introduzca un retardo respecto a la señal original.

Una vez obtenidas las matrices Módulo *delay* y Fase *delay,* se multiplica Módulo *delay* por el parámetro *g* y Fase *delay* por el parámetro *tn* (los parámetros *g* y *tn* se corresponden con los parámetros *Ganancia* y *Tono* del *script* de *Matlab* correspondiente). De esta manera, mediante el parámetro *g* se eleva o disminuye el nivel de volumen que tendrá la señal retardada respecto a la señal original, y mediante el parámetro *tn* se modifican ligeramente las frecuencias instantáneas de la señal. Así se puede obtener una repetición pura de la señal o una repetición de la señal desplazada en frecuencia, creando un efecto eco ligeramente diferente del concepto de eco tradicional.

El último paso del algoritmo es realizar la síntesis aditiva para la señal original y la señal *delay* y sumar ambas señales, de manera que el resultado es que se escucha la señal original junto a la señal retardada.

3.1.2.2. Chorus

El efecto *Chorus* consiste en repetir la señal de audio original tantas veces como se desee aplicándole cada vez un tiempo de *delay* aleatorio que varía entre 10 y 25 ms. A continuación se forma una nueva señal de audio formada por la suma de la señal original más todas las señales generadas.

La forma de implementar el efecto *Chorus* a partir del algoritmo CWAS resulta muy sencilla:

- Primero se aplica la CWAS al sonido original obteniendo la matriz Parcial, que contiene la información de todos los parciales.

- A partir de la matriz Parcial se obtienen las matrices Módulo y Fase.

- Se establece el número de voces que se repetirá el sonido original (*n*). Aunque no hay límite en el número de voces, su incremento ralentiza la ejecución del algoritmo y no supone una mejora significativa en los resultados. Por tanto, se recomienda un número de voces comprendido entre 2 y 5, lo que supone un buen compromiso ente velocidad y calidad.

- Sobre las matrices Módulo y Fase se aplica la estructura DZP para obtener tantas matrices Módulo *delay* y Fase *delay* como voces se desean para el efecto. En este caso, la estructura DZP tiene la particularidad de que el *zero-padding* que se aplica para generar las matrices Módulo y Fase adaptadas debe tener el número de *samples* del máximo *delay* que se puede obtener en las señales repetidas, es decir, 25 ms. A cada una de las matrices Módulo *delay* y Fase *delay* se les aplica el *zero-padding* de forma normal al inicio y otro *zero-padding* al final que debe tener la duración en *samples* indicada en la ecuación 3.3.

$$samples_{final} = samples_{25ms} - samples_{delay} \qquad \text{(ec. 3.3)}$$

  de manera que todas las matrices deben tener la misma duración para no tener problemas al operar con ellas en *Matlab*.

- Para realizar el *zero-padding*, a cada nueva voz se le aplica un *delay* aletorio, que se calcula utilizando la ecuación 3.4.

$$samples_{delay(aleatorio)} = (samples_{25ms} - samples_{10ms}) \times a + samples_{10ms} \qquad \text{(ec. 3.4)}$$

  donde *a* es un número aleatorio entre 0 y 1 y $samples_{25ms}$ y $samples_{10ms}$ son el número de *samples* que corresponde a un *delay* de 25 y 10 ms respectivamente y que se calculan con la ecuación 3.2.

- Se multiplica la matriz Fase correspondiente a las voces que se añaden por un parámetro *tn* aleatorio (parámetro *Tono* del *script* correspondiente), de manera que todas las voces no suenan exactamente igual y se proporciona un mayor realismo al efecto.

- Una vez se tienen todas las matrices, se realiza la síntesis aditiva para generar el sonido sintético original y los sonidos sintéticos de cada voz, de manera que la señal de audio de salida del efecto es la suma de todos los sonidos generados.

3.1.2.3. Vibrato

El *Vibrato* es una variación periódica de la frecuencia de un sonido, por lo que si mediante al algoritmo CWAS se divide el sonido en una matriz Módulo y una matriz Fase (directamente relacionada con la frecuencia instantánea del sonido), aplicando una variación a la matriz Fase del sonido se puede conseguir el efecto *Vibrato* de una forma más sencilla que mediante los métodos tradicionales.

En la Figura 3.6 se muestra, de forma esquemática, cómo se implementa el efecto *Vibrato* a partir del algoritmo CWAS.

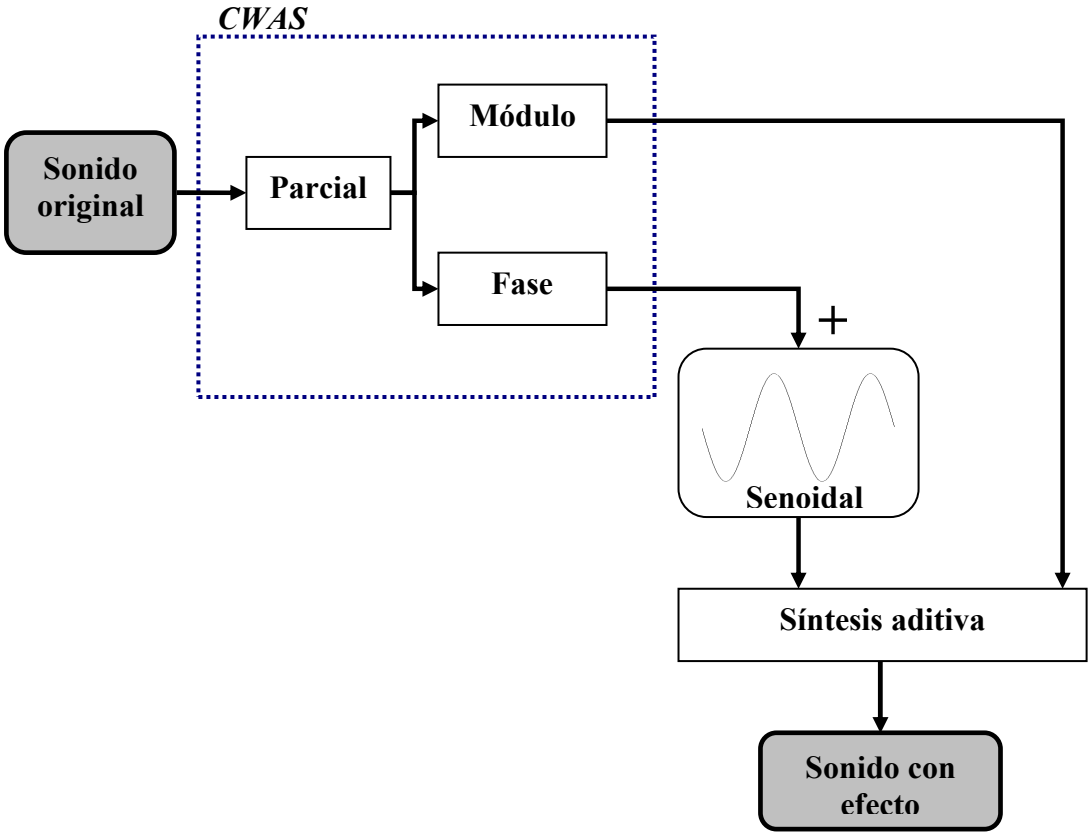


Figura 3.6. Diagrama del efecto *Vibrato*.

La forma de implementar el efecto *Vibrato* a partir del algoritmo CWAS consiste en lo siguiente:

- Se somete a la señal de audio al análisis mediante la CWAS para obtener las matrices Módulo y Fase. La matriz Módulo se mantendrá inalterada en el proceso.

- Se genera la matriz Senoidal de tamaño MxN, donde M es la longitud en samples del sonido original y N es el número de parciales que forman la señal original. La matriz Senoidal contiene, para cada parcial y para la extensión en *samples* del sonido, una onda sinusoidal de baja frecuencia (entre 5 y 14 Hz), que se corresponde con la ecuación 3.5.

$$y(t) = A \cdot \sin(2\pi \cdot f \cdot t) \qquad \text{(ec. 3.5)}$$

donde A es la amplitud de la onda sinusoidal y marca la profundidad en tono que tiene el *Vibrato* (a mayor valor de A más se notará la variación en frecuencia) y f es la frecuencia de dicha onda sinusoidal, que oscila entre 5 y 14 Hz a elegir por el usuario.

- Una vez generada la matriz Senoidal, se suman las matrices Fase y Senoidal.

$$Fase_{final} = Fase + Senoidal \qquad \text{(ec. 3.6)}$$

- Por último, se realiza la síntesis aditiva entre la nueva matriz Fase y la matriz Módulo que, como se ha comentado anteriormente, se mantiene inalterada durante todo el proceso.

3.2. TIME-STRETCHING

Hay ocasiones en las que resulta necesario adaptar una señal de audio ya grabada a unas exigencias de duración temporal o de velocidad de reproducción dadas. Esto puede conseguirse fácilmente ya que un sonido en formato digital se puede reproducir a distintas velocidades cambiando la frecuencia de muestreo con la que trabaja el reproductor de audio. Sin embargo, aunque con este tipo de aplicaciones se realiza un cambio en la velocidad de reproducción, se cambia también el tono del sonido reproducido.

El efecto de *Time-stretching* consiste en modificar la duración de un sonido y su velocidad de reproducción sin alterar el tono del audio original. La relación entre la duración de un sonido y su velocidad de reproducción es inversamente proporcional, es decir, si se aumenta la duración temporal del sonido se disminuye su velocidad de reproducción y viceversa.

3.2.1. PROGRAMACIÓN TRADICIONAL

La implementación del *Time-stretching* mediante métodos basados en la STFT se realiza de la siguiente manera:

- Primero se divide la señal de audio de entrada en varios fragmentos.

- En cada fragmento, se obtiene la representación de módulos y fases cada *n1 samples* a través de la STFT.

- A continuación se calcula una secuencia de *n2 samples* de la señal de salida interpolando los valores de los módulos y calculando el valor de las fases para cada nuevo *sample,* de manera que la derivada de la fase de salida (la frecuencia) sea igual a la derivada de la fase de entrada.

- Finalmente se reconstruye la señal de audio de salida mediante la transformada inversa de Fourier (IFFT).

Básicamente lo que se hace es interpolar los módulos para los *n2 samples* y recalcular las fases para mantener la frecuencia constante.

La duración de la señal de salida queda multiplicada por el factor *time-strech*, $\delta_{t\text{-}s}$ (ecuación 3.8).

$$\delta_{t-s} = \frac{n2}{n1} \qquad \text{(ec. 3.7)}$$

$$t_{out} = t_{in} \times \delta_{t-s} \qquad \text{(ec. 3.8)}$$

donde t_{in} es el tiempo que dura el sonido original y t_{out} es el tiempo que dura el sonido que se obtiene tras aplicar el *Time-stretching*.

En cambio, la velocidad de reproducción queda multiplicada por la inversa del factor *time-stretch* (ecuación 3.9).

$$v_{out} = v_{in} \times \frac{1}{\delta_{t-s}} \qquad \text{(ec. 3.9)}$$

3.2.2. PROGRAMACIÓN CWAS

La implementación del efecto *Time-stretching* mediante el uso de las técnicas tradicionales conlleva más complejidad que los efectos anteriores. De igual modo, su implementación mediante el uso de la CWAS también resulta más complicada.

Para la implementación CWAS de este efecto se ha definido el parámetro γ (parámetro *Incremento* en el *script* de *Matlab* correspondiente), que es el parámetro que marca la nueva duración temporal del sonido. Este parámetro se ha definido de manera que puede tomar valores positivos o negativos. Los valores de γ positivos hacen que la señal aumente su duración temporal, mientras que los valores negativos hacen que la duración temporal disminuya.

La relación existente entre el parámetro γ y la duración temporal del sonido se muestra en la ecuación 3.10.

$$duración = \begin{cases} duración_{\,0} \times \gamma & \text{si } \gamma > 0 \\ duración_{\,0} \times \dfrac{1}{|\gamma|} & \text{si } \gamma < 0 \end{cases} \qquad \text{(ec. 3.10)}$$

Se define también el parámetro ρ (parámetro *Factor* en el *script* de *Matlab*) según la ecuación 3.11.

$$\rho = \frac{duración}{duración_0} \qquad \text{(ec 3.11)}$$

Se transforma ρ en un número racional, de manera que:

$$\rho = \frac{n1}{n2} \qquad \text{(ec. 3.12)}$$

donde *n1* será el número de *samples* de *upsampling* y *n2* el número de *samples* de *downsampling*.

Básicamente, lo que hace el efecto *Time-stretching* es aumentar el número de muestras en un factor *n1* y disminuirlo en un factor *n2*. Esto significa que cada *sample* de la señal original se repite *n1* veces (*upsampling*) y, una vez hecho esto, se selecciona un *sample* de entre cada *n2* para el sonido final (*downsampling*).

En la implementación a partir del algoritmo CWAS debe aplicarse el *upsampling* y el *downsampling* tanto sobre la matriz Módulo como sobre la matriz Fase. Para aplicar este método sobre la matriz Módulo, se repetirá el proceso descrito anteriormente para cada parcial de la matriz Módulo (para cada columna de la matriz). Sin embargo, a la hora de implementarlo sobre la matriz Fase, este proceso debe aplicarse sobre la derivada de la matriz y luego integrar para obtener de nuevo la matriz Fase. Igual que en el caso de la matriz Módulo, el proceso se repite para cada columna de la matriz Fase derivada. Si el proceso se aplicase directamente sobre la matriz Fase, aparecerían discontinuidades bruscas que se traducirían en artefactos no deseados al realizar la síntesis aditiva.

En la Figura 3.7 se ha representado de forma esquemática el proceso que acaba de describirse.

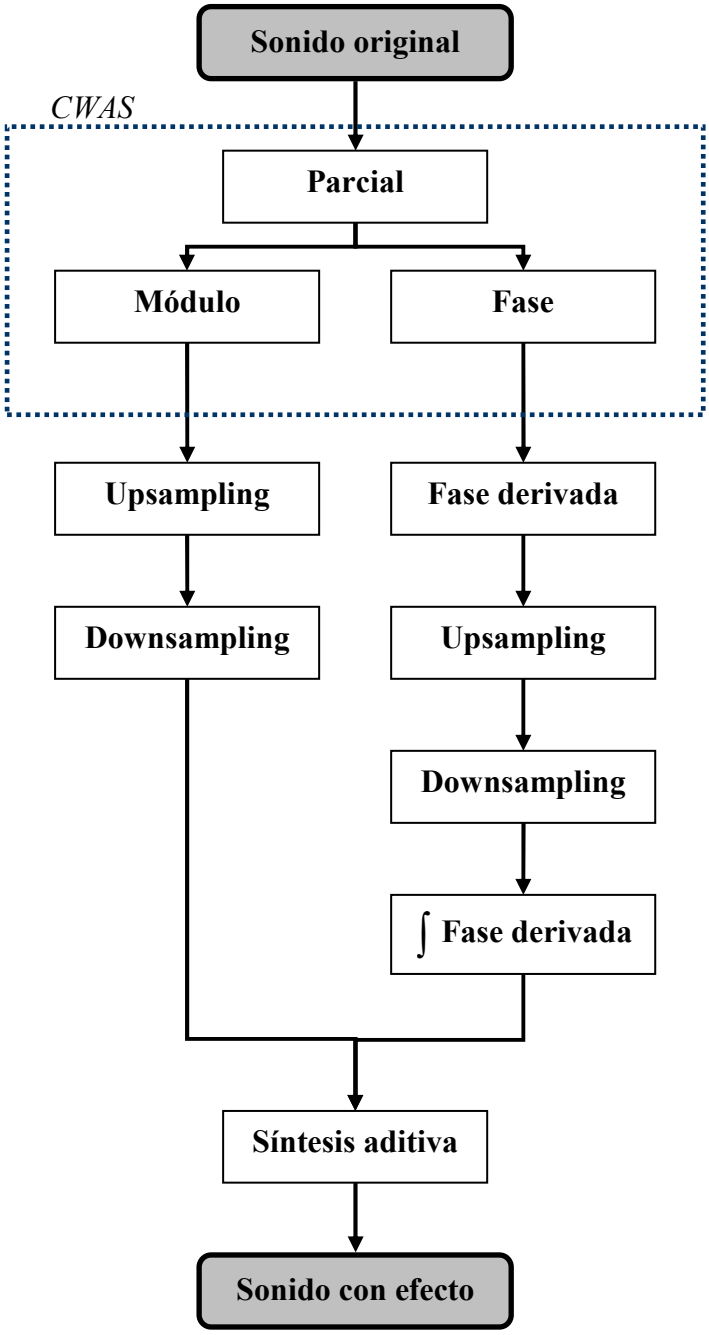


Figura 3.7. Diagrama del efecto *Time-stretching*.

3.3. ROBOTIZACIÓN

El efecto de *Robotización* es un efecto que se aplica sobre voces habladas y consiste en hacer que una voz humana suene como la voz de un robot.

3.3.1. PROGRAMACIÓN TRADICIONAL

La implementación del efecto de *Robotización* mediante las técnicas basadas en la STFT es muy sencilla y consiste en lo siguiente:

- Se obtiene un fragmento de señal de *n samples* comenzando en el primer *sample* del sonido y se multiplica dicho fragmento por una ventana de *hanning* de tamaño *n*.

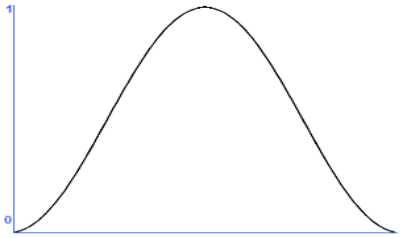


Figura 3.8. Ventana de *hanning*.

- Se aplica la FFT al fragmento anterior y se separan módulo y fase.

- Se aplica la transformada inversa (IFFT) al fragmento, dando como valores de módulo los obtenidos en el paso anterior y a las fases se les da el valor cero.

- Se multiplica el resultado de la IFFT por una ventana de *hanning* de tamaño *n*.

- Una vez reconstruida la señal de este fragmento se guarda en la señal de salida.

- Se vuelve a repetir el procedimiento comenzando en el siguiente *sample*, y así sucesivamente hasta que se barre toda la señal.

El resultado es una voz robotizada, es decir, un sonido producido con un tono fijo.

3.3.2. PROGRAMACIÓN CWAS

Como se ha comentado anteriormente, el efecto de *Robotización* pretende que una voz hablada suene similar a la voz de un robot. En otras palabras, lo que hace dicho efecto es aplicar un tono fijo a la voz.

Los pasos a seguir para realizar la robotización CWAS son los siguientes:

- Obtener las matrices Módulo y Fase a partir de la aplicación del algoritmo CWAS. La matriz Módulo se mantendrá inalterada en el proceso.

- Se elige una frecuencia base para el sonido y, a partir de dicha frecuencia, se generan las fases para la extensión total del sonido. La frecuencia base se ha definido con el parámetro ε (parámetro *Frecuencia* en el *script* de *Matlab* correspondiente). La forma de generar la fase viene dada por la ecuación 3.13.

$$\varphi_g(i) = \frac{2\pi \times \varepsilon}{Fs} \times i \qquad \text{(ec. 3.13)}$$

donde φ_g es la fase generada, *Fs* es la frecuencia de muestreo e *i* es el número de *sample* (desde 1 hasta *n* = número de *samples* totales del sonido).

- En la matriz Fase se reemplazan los 15 primeros parciales (los de mayor contenido energético) por la fase que se ha generado de acuerdo con la frecuencia base. Las fases del resto de parciales no se modifican ya que se ha comprobado que aumentan el tiempo de cálculo y no aportan diferencias significativas en el resultado final.

- Se realiza la síntesis aditiva para generar el sonido a partir de la matriz Módulo y la nueva matriz Fase.

En la Figura 3.9 se muestra de forma esquemática el proceso que acaba de detallarse.

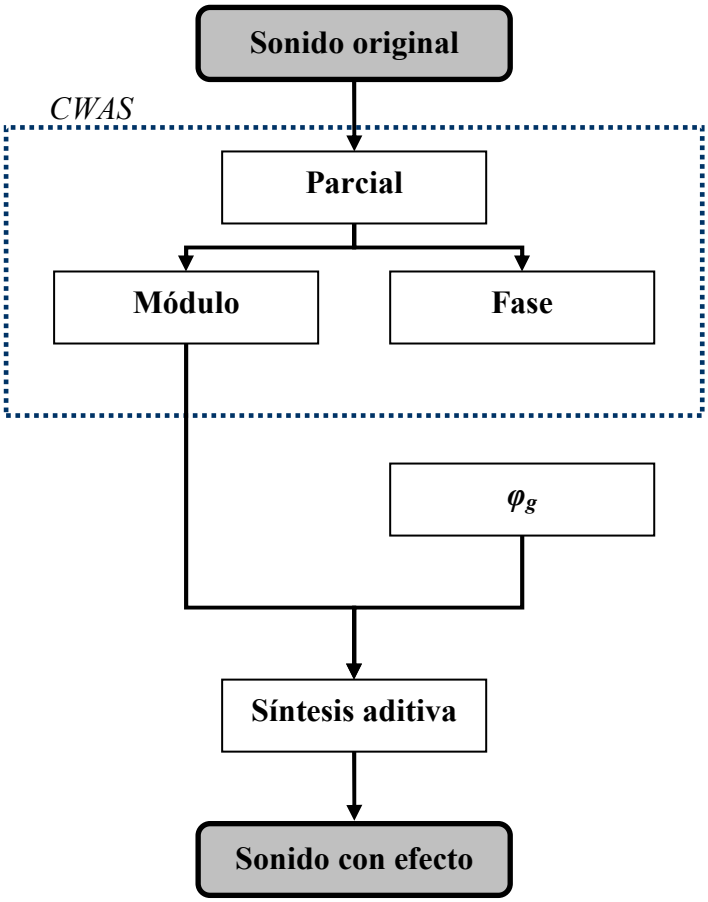


Figura 3.9. Diagrama del efecto *Robotización*.

3.4. DENOISING

Normalmente, en las grabaciones acústicas aparecen ruidos no deseados que resultan molestos y ensucian la grabación. Por ello, la aplicación de un efecto que limpie la señal y filtre la mayor cantidad posible de ese ruido indeseado resulta de gran utilidad. Este efecto se conoce con el nombre de *Denoising*.

3.4.1. PROGRAMACIÓN TRADICIONAL

El objetivo del efecto *Denoising* es, como ya se ha comentado, eliminar el ruido de una señal. Su implementación a partir de técnicas tradicionales consiste en lo siguiente:

- En primer lugar se obtienen los módulos y fases del sonido mediante la STFT.
- Las fases se mantienen inalteradas y los módulos se tratan conservando sus valores más altos y atenuando sus valores más bajos.
- Se reconstruye la señal a través de las IFFT usando los nuevos módulos y las fases originales.

Para atenuar el ruido se hace uso de una función no lineal que permite suavizar los valores bajos de la señal sin alterar los valores más altos. La función utilizada para ello se muestra en la ecuación 3.14.

$$f(x) = \frac{x}{x + c} \qquad \text{(ec. 3.14)}$$

Para atenuar los módulos, éstos se multiplican por la función *f (x)*, donde *x* es el valor del módulo en cada *sample* y *c* es el factor de atenuación, al que se le da el valor de 0,01.

A continuación se muestra un ejemplo numérico acerca del funcionamiento del factor de atenuación sobre la función *f* (*x*):

Con $x = 0{,}2 \rightarrow x = 0{,}2 \times \frac{0{,}2}{0{,}2 + 0{,}01} = 0{,}082$ La señal se atenúa en un 59%.

Con $x = 10 \rightarrow x = 10 \times \frac{10}{10 + 0{,}01} = 9{,}99$ La señal se atenúa en un 0,1%.

Siempre que la magnitud del ruido sea de un orden mucho menor a la magnitud de la señal grabada se eliminará gran parte del ruido, pero si la magnitud del ruido es del orden de magnitud de la señal grabada, este algoritmo no obtendrá buenos resultados.

Para entender mejor cómo funciona el efecto *Denoising*, en la Figura 3.10 se muestra el efecto exagerado que se produce con la aplicación de una función de atenuación *f (x)*.

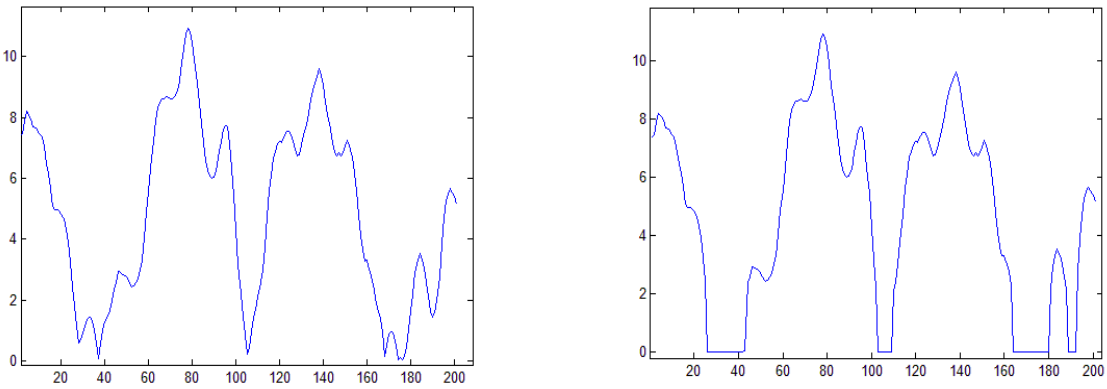


Figura 3.10.A la izquierda se muestra una función *f(x)* con ruido y
a la derecha se observa la misma función *f(x)* después de eliminar el ruido.

3.4.2. PROGRAMACIÓN CWAS

La implementación del efecto *Denoising* mediante el método CWAS es muy similar al método tradicional, con la diferencia de que el algoritmo CWAS debe actuar sobre los módulos de cada parcial del sonido, es decir, sobre cada columna de la matriz Módulo.

Como se puede apreciar en la Figura 3.11, el proceso que se sigue para implementar el efecto *Denoising* según la programación CWAS es muy sencillo. Consiste en obtener las matrices Módulo y Fase a partir del algoritmo CWAS y, una vez hecho esto, se mantiene la matriz Fase inalterada y se somete a la matriz Módulo a un filtrado.

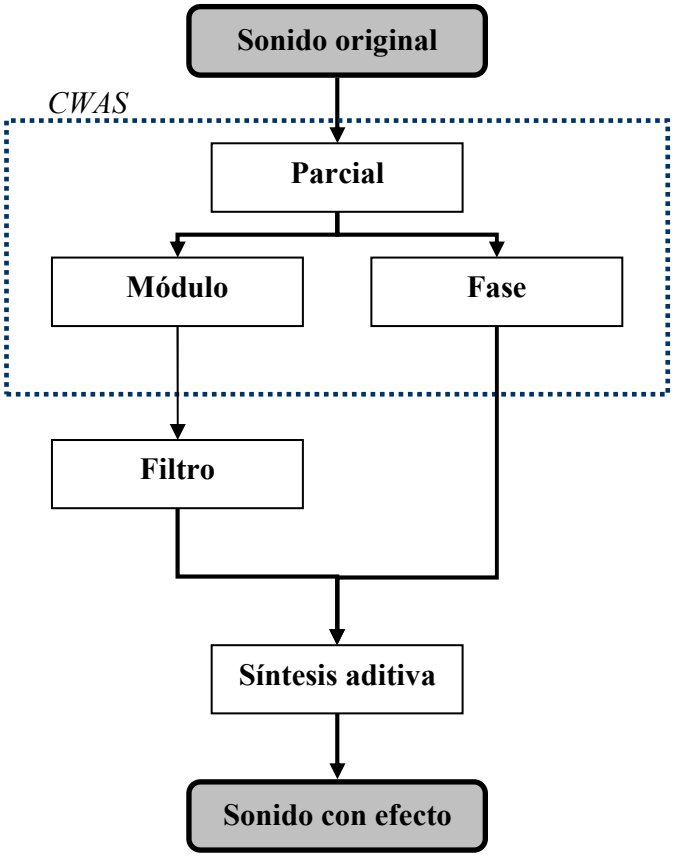


Figura 3.11. Diagrama del efecto *Denoising*.

El filtrado de la matriz Módulo consiste en recorrer dicha matriz elemento por elemento, aplicándole a cada uno la ecuación 3.15.

$$y(i,j) = x(i,j) \times \frac{x(i,j)}{x(i,j)+c} \qquad \text{(ec. 3.15)}$$

donde c es el factor de atenuación (c=0,01) que, como se ha explicado en el apartado anterior, se usa para suavizar los valores cuyo módulo tienen un valor bajo (ruido), *y(i, j)* es el nuevo valor que tendrá cada elemento de la matriz Módulo y *x(i, j)* es el valor original de cada elemento de la matriz Módulo.

$$\begin{cases} i = 1,...,m & (m \text{ es el número de } \textit{samples} \text{ del sonido}) \\ j = 1,...,n & (n \text{ es el número de parciales del sonido}) \end{cases}$$

Generalmente gran parte del ruido que se origina en una grabación o durante una actuación en vivo con instrumentos eléctricos tiene su origen en la frecuencia de la red eléctrica. Esta frecuencia tiene un valor de 50 o 60 Hz dependiendo de los países. Por tanto, la última parte de este filtro se centra en encontrar las frecuencias medias de cada parcial del sonido y eliminar los módulos de aquellos parciales cuyas frecuencias correspondan a 50 o 60 Hz o estén próximas a estos valores.

Para calcular las frecuencias medias de cada parcial es necesario obtener primero las frecuencias instantáneas de cada *sample*, para lo cual debe calcularse la derivada de la matriz Fase. Haciendo la media aritmética de las frecuencias instantáneas de cada parcial se obtiene su frecuencia media.

Habitualmente se utiliza la ecuación 3.16 para calcular las frecuencias instantáneas.

$$f_{ins\,\tan\,tánea} = \frac{1}{2\pi} \times \frac{d}{dt}\varphi(t) \qquad \text{(ec. 3.16)}$$

Sin embargo, debido a que las derivadas en *Matlab* se implementan como una resta (mediante el comando DIFF), para el algoritmo desarrollado en este proyecto se ha utilizado la ecuación 3.17, que es una forma más simple de obtener las frecuencias instantáneas.

$$f_{ins\,\tan\,tánea} = \frac{Fs}{2\pi} \times DIFF(Fase) \qquad \text{(ec. 3.17)}$$

donde *Fs* es la frecuencia de muestreo y Fase es la matriz Fase. De la ecuación 3.17 se obtendrá una matriz con las frecuencias instantáneas de cada parcial y cada *sample* del sonido.

Para acabar con el proceso se realiza la síntesis aditiva de las matrices Módulo (una vez filtrada) y Fase, generándose la señal de audio con el efecto aplicado.

3.5. PITCH-SHIFTING

El *Pitch-shifting* es un efecto que consiste en cambiar el tono de una grabación sin modificar su duración temporal. Para ello se multiplican todas las frecuencias por un factor de transposición *k*.

3.5.1. PROGRAMACIÓN TRADICIONAL

Una manera sencilla de conseguir el efecto *Pitch-shifting* es transponer todas las frecuencias instantáneas sin modificar los módulos. Para ello, su implementación a partir del método tradicional sigue los siguientes pasos:

- En primer lugar se obtienen el módulo y la fase de cada *sample* mediante la FFT.

- Se calcula el incremento de fase entre cada *sample* ($\Delta\varphi$) y se multiplica por el factor de transposición *k*. La nueva fase se integra de acuerdo con la ecuación 3.18.

$$\varphi(n+1) = \varphi(n) + \Delta\varphi \times k \qquad \text{(ec. 3.18)}$$

- Se reconstruye la señal como una suma de sinusoides.

Las frecuencias instantáneas de la señal de salida quedan multiplicadas por el factor de transposición *k*.

$$f = f_0 \times k \qquad \text{(ec. 3.19)}$$

3.5.2. PROGRAMACIÓN CWAS

Gracias a las ventajas que brinda el uso del algoritmo CWAS como punto de partida para la programación de efectos, el *Pitch-shifting* puede implementarse de forma sencilla y conservando con mayor fidelidad las características del sonido.

Generalmente, a la hora de desplazar un sonido en frecuencia, en el argot musical se habla de transportar el sonido un número de semitonos hacia arriba (sonido más agudo o de mayor frecuencia) o hacia abajo (sonido más grave o de menor frecuencia). Por tanto, a la hora de implementar un *Pitch-shifting* será mucho más práctico hablar de los semitonos que se quiere desplazar el sonido y establecer una relación entre el número de semitonos y la frecuencia.

El semitono es la mínima división de altura utilizada en la música occidental. Matemáticamente, el semitono determina una relación de frecuencias de $2^{1/12}$ (aproximadamente 1,059) [4]. Por consiguiente, cada vez que se sube un semitono, la frecuencia resultante se multiplica por 1,059.

En la Tabla 3.2 se muestra la correspondencia entre todas las notas musicales y su equivalente en frecuencia.

Tabla 3.2. Relación entre notas musicales y frecuencias.

Nota	Freq. (Hz)	Nota	Freq.(Hz)	Nota	Freq.(Hz)
La 1	27.500	La 4	220.000	La 7	1760.000
La# 1	29.135	La# 4	233.082	La# 7	1864.655
Si 1	30.868	Si 4	246.942	Si 7	1975.533
Do 2	32.703	Do 5	261.626	Do 8	2093.005
Do# 2	34.648	Do# 5	277.183	Do# 8	2217.461
Re 2	36.708	Re 5	293.665	Re 8	2349.318
Re# 2	38.891	Re# 5	311.127	Re# 8	2489.016
Mi 2	41.203	Mi 5	329.628	Mi 8	2637.021
Fa 2	43.654	Fa 5	349.228	Fa 8	2793.826
Fa# 2	46.249	Fa# 5	369.994	Fa# 8	2959.956
Sol 2	48.999	Sol 5	391.995	Sol 8	3135.964
Sol# 2	51.913	Sol# 5	415.305	Sol# 8	3322.438
La 2	55.000	La 5	440.000	La 8	3520.000
La# 2	58.270	La# 5	466.164	La# 8	3729.310
Si 2	61.735	Si 5	493.883	Si 8	3951.066
Do 3	65.406	Do 6	523.251	Do 9	4186.009
Do# 3	69.296	Do# 6	554.365	Do# 9	4434.922
Re 3	73.416	Re 6	587.330	Re 9	4698.637
Re# 3	77.782	Re# 6	622.254	Re# 9	4978.032
Mi 3	82.407	Mi 6	659.255	Mi 9	5274.042
Fa 3	87.307	Fa 6	698.457	Fa 9	5587.652
Fa# 3	92.499	Fa# 6	739.989	Fa# 9	5919.912
Sol 3	97.999	Sol 6	783.991	Sol 9	6271.928
Sol# 3	103.826	Sol# 6	830.609	Sol# 9	6644.876
La 3	110.000	La 6	880.000	La 9	7040.000
La# 3	116.541	La# 6	932.328	La# 9	7458.620
Si 3	123.471	Si 6	987.767	Si 9	7902.133
Do 4	130.813	Do 7	1046.502	Do 10	8372.019
Do# 4	138.591	Do# 7	1108.731	Do# 10	8869.845
Re 4	146.832	Re 7	1174.659	Re 10	9397.273
Re# 4	155.564	Re# 7	1244.508	Re# 10	9956.064
Mi 4	164.814	Mi 7	1318.510	Mi 10	10548.083
Fa 4	174.614	Fa 7	1396.913	Fa 10	11175.305
Fa# 4	184.997	Fa# 7	1479.978	Fa# 10	11839.823
Sol 4	195.998	Sol 7	1567.982	Sol 10	12543.855
Sol# 4	207.652	Sol# 7	1661.219	Sol# 10	13289.752

En el algoritmo CWAS que se ha implementado para el *Pitch-shifting* el usuario debe introducir el número de semitonos que quiere desplazar el sonido. Si se quiere desplazar el sonido n semitonos hacia arriba en frecuencia, en la variable semitonos se introducirá $+n$. Si por el contrario se quiere desplazar el sonido hacia abajo en frecuencia se introducirá $-n$.

El esquema de la implementación del efecto *Pitch-shifting* a partir del algoritmo CWAS puede apreciarse en la Figura 3.12.

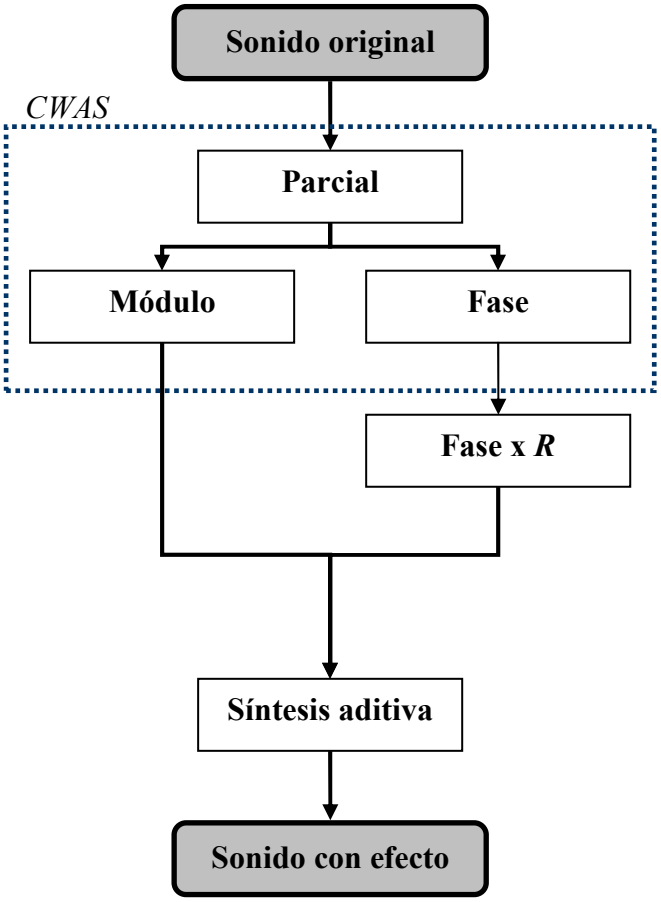


Figura 3.12. Diagrama del efecto *Pitch-shifting*.

Básicamente el *Pitch-shifting* consiste en multiplicar las frecuencias instantáneas por el factor *R* (parámetro *Ratio* en el *script* de *Matlab* correspondiente), definido en la ecuación 3.20. Al multiplicar las frecuencias instantáneas por *R*, se multiplican también las fases. Por tanto, para implementar el efecto basta con multiplicar la matriz Fase por el parámetro *R*. Una vez hecho esto, se realiza la síntesis aditiva y se obtiene el sonido de salida con el efecto aplicado.

$$R = \begin{cases} 1.095^{\,semitonos} & \text{si } semitonos > 0 \\ \dfrac{1}{1.095^{\,|semitonos|}} & \text{si } semitonos < 0 \end{cases} \qquad \text{(ec. 3.20)}$$

donde *semitonos* es el número de semitonos que se quiere desplazar la señal.

3.6. PITCH-SHIFTING MEJORADO

Como se acaba de comentar, el *Pitch-shifting* es un efecto que consiste en cambiar el tono de una grabación. Sin embargo, este efecto sólo funciona bien cuando el sonido original se desplaza unos pocos semitonos arriba o abajo. En cambio, cuando el desplazamiento en semitonos es más pronunciado el resultado es un sonido poco

natural. Por ejemplo, si se realiza un *Pitch-shif* de una octava (12 semitonos) por encima del tono original sobre un fragmento de voz humana, bien sea hablada o cantada, se produce un sonido con un efecto "*Donald duck*" que no suena igual que cuando una persona habla o canta el mismo fragmento una octava por encima de la grabación original.

El efecto *Pitch-shifting mejorado* es una evolución del efecto anterior pensado para su aplicación sobre instrumentos musicales o voces humanas, siempre que en ambos casos se conozca la respuesta real a lo largo de todo su rango acústico y se disponga de varias muestras de sonido dentro del mismo. De esta forma, es posible predecir las características sonoras de todas las notas que pueden ejecutarse, adaptando el sonido transportado al sonido real. El objetivo es conseguir saltos de un mayor número de semitonos de forma que el sonido tenga una respuesta similar a la que tendría el instrumento real. Así, conociendo el sonido de una única nota ejecutada por un instrumento podría generarse toda la respuesta frecuencial reproducible por dicho instrumento. En otras palabras, lo que quiere conseguirse es, por ejemplo, transportar la grabación de un Do3 a un Do4 y que ese Do4 suene lo más próximo posible al Do4 real ejecutado por el instrumento o voz en cuestión.

No se conoce la implementación de este efecto mediante técnicas tradicionales.

3.6.1. PROGRAMACIÓN CWAS

La idea fundamental del *Pitch-shifting mejorado* es muy sencilla. Si se analizan los escalogramas (respuesta en frecuencia) de dos notas distintas, por ejemplo un Do3 y un Do4, puede apreciarse que si se desplaza la primera nota para que suene como la segunda, el escalograma resultante es ligeramente diferente del escalograma original de ese Do4. Esto significa que, aunque suene un Do4, sonará ligeramente diferente del sonido del Do4 real.

Para implementar este efecto mediante el algoritmo CWAS se necesita una nota de muestra o nota base y, además, será necesario conocer la respuesta en frecuencia del instrumento o de la voz a lo largo de todo su rango acústico.

La aplicación de este método sería especialmente útil en sintetizadores (instrumentos electrónicos de teclas que simulan los sonidos de instrumentos reales con más o menos exactitud) ya que a partir de un único sonido se pueden generar todas las notas que da el instrumento. Esto supondría un ahorro de memoria para el sintetizador.

Para implementar el efecto a partir del algoritmo CWAS se ha elegido una nota base y varias notas contenidas en un rango máximo de $\pm$ 1 octava respecto a la nota base. El método es extrapolable a todo el rango acústico del instrumento siempre que existan grabaciones del mismo.

En la Figura 3.13 se han representado las notas utilizadas para implementar el efecto *Pitch-shifting mejorado* en las teclas de un piano.

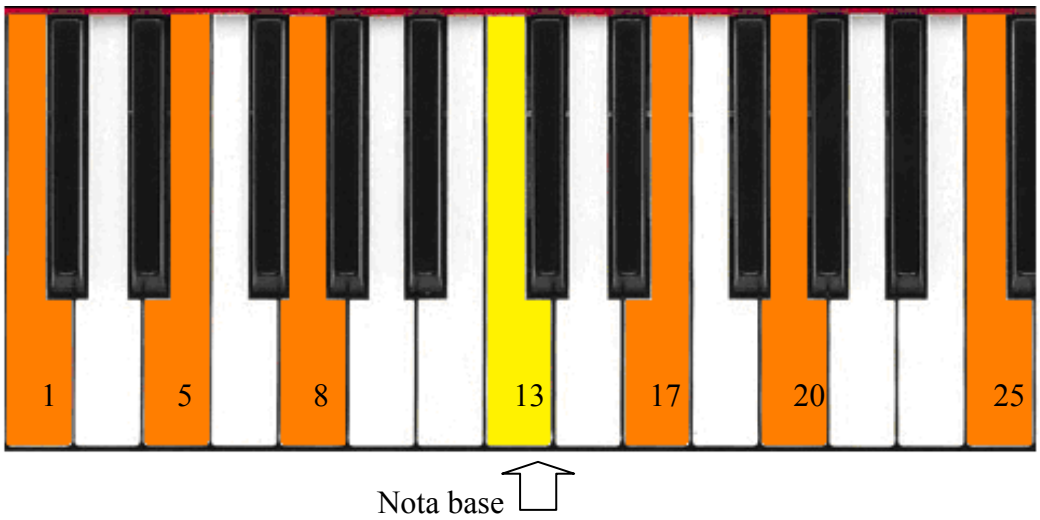


Figura 3.13. Notas conocidas para implementar el efecto *Pitch-shifting mejorado*.

Como puede apreciarse en la Figura 3.13, el efecto *Pitch-shifting mejorado* se ha implementado en un rango de 2 octavas completas con sus 25 semitonos. Dentro de las 2 octavas, la nota base es el segundo Do (semitono 13). Para implementar el efecto se han usado las notas cuyos números de semitono dentro de las 2 octavas son 1, 5, 8, 13, 17, 20 y 25.

A estas notas se les aplica el algoritmo CWAS con el objetivo de guardar sus escalogramas y conocer así el comportamiento en frecuencias de cada una de ellas.

Una vez hecho esto, se realiza una interpolación de los escalogramas guardados para conocer, de forma aproximada, cómo debe ser el escalograma de cada uno de los 25 semitonos que forman las 2 octavas. De esta forma se conocen los escalogramas de todas las notas y se almacenan en una matriz llamada Interpolación, que será una matriz con 25 columnas. Cada columna corresponde al escalograma interpolado de cada una de las 25 notas musicales que forman las dos octavas.

A la hora de implementar el efecto *Pitch-shifting mejorado* se siguen los siguientes pasos:

- En primer lugar se aplica el algoritmo CWAS a la nota base y se obtienen sus matrices Módulo y Fase.

- Se establece el parámetro R, por el que se multiplicarán las frecuencias instantáneas de cada parcial de la nota base. El parámetro R se calcula de la misma manera que se hacía en el *Pitch-shifting* (ecuación 3.20).

- Se calcula el vector α para la nota base (vector *imodulemedian* en el *script* de *Matlab* correspondiente), cuyos elementos son la media de los módulos de cada parcial de esa nota.

- Se calculan las frecuencias medias de cada parcial y se almacenan en un vector.

- Se buscan los escalogramas de la nota base y de la nota a la que se quiere llegar (nota destino) en la matriz Interpolación.

- Se buscan todos los picos del escalograma de la nota destino y se almacenan las frecuencias a las que se produce cada uno. Cada pico del escalograma corresponde a un parcial importante en energía (armónicos importantes de la nota).

- Se multiplican la matriz Fase y el vector de frecuencias medias por *R*. De esta manera se desplaza la nota base a la nota destino.

- Para cada pico del escalograma de la nota destino, se buscan la nueva frecuencia media a la que corresponde y su parcial asociado.

- Se compara el valor del elemento del vector α correspondiente a ese parcial con la energía que tiene ese pico del escalograma destino. Se aumenta o disminuye el valor de los módulos del parcial para que ese armónico de la nota destino esté igual de realzado o atenuado que en la nota destino real.

- Se realiza la síntesis aditiva, obteniéndose el sonido de la nota base.

La etapa de localización de los picos del escalograma resulta un poco más compleja que el resto de pasos, por lo que se explica con más detalle a continuación.

En la Figura 3.14 aparece el ejemplo de un escalograma que pertenece a una nota ejecutada en una guitarra.

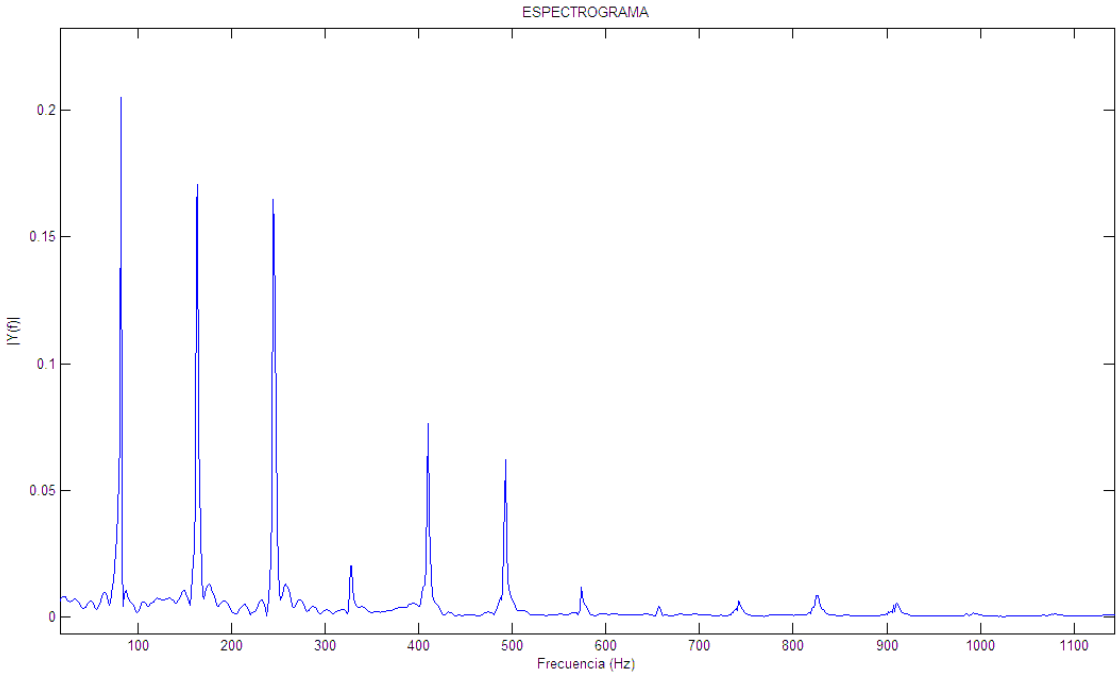


Figura 3.14. Escalograma de una nota ejecutada por una guitarra.

Como se puede apreciar en la Figura 3.14, el escalograma contiene una serie de picos que se corresponden con la frecuencia fundamental de una nota y con los armónicos o múltiplos de dicha frecuencia.

La parte del algoritmo que localiza los picos del escalograma destino realiza los siguientes pasos:

- Se localiza el máximo total del escalograma, que será un pico del mismo, y se almacenan sus valores de frecuencia y energía.
- Se eliminan tanto el pico localizado (valor máximo) como su zona adyacente para que al repetir el proceso no se localice un pico falso.
- Se repite el proceso hasta que se localizan los 30 picos más importantes en energía del escalograma.

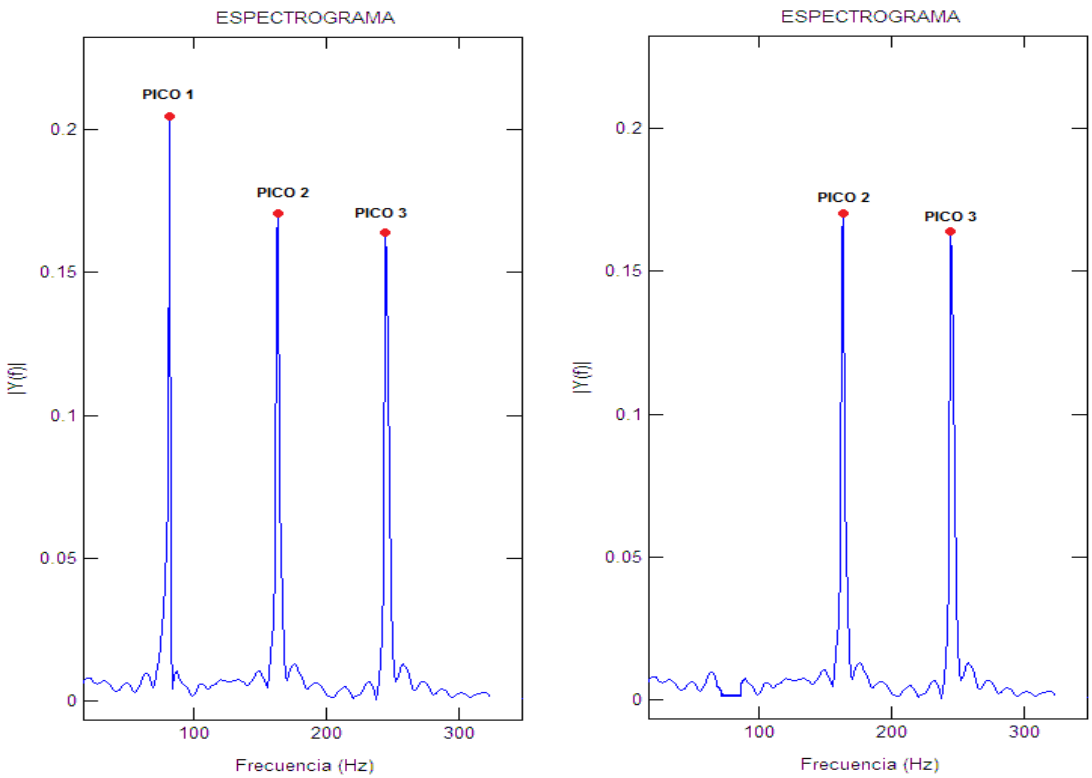


Figura 3.15. Proceso de localización y eliminación de picos.

En la Figura 3.15 de la izquierda se muestra un tramo del escalograma de la Figura 3.14 en el que aparecen los tres primeros picos. En la Figura 3.15 de la derecha se ha eliminado el primer pico junto con la zona que está a su alrededor.

3.7. SUSTAIN

Para definir el efecto *Sustain* es necesario analizar previamente el ciclo de vida de un sonido. El ciclo de vida de un sonido se adapta a una estructura de envolventes de volumen que dan forma al sonido. La más común de estas estructuras es la que recibe el nombre de ADSR [4]. Según esta estructura, el ciclo de vida de un sonido ejecutado por un instrumento musical se divide en cuatro secciones:

- *Attack* (ataque): Cuando se ejecuta una nota o acorde con un instrumento musical se produce un incremento rápido de la señal hasta alcanzar un valor máximo o pico. La región *Attack* es aquélla comprendida desde el inicio del sonido hasta que se alcanza dicho máximo. Generalmente es una región de poca duración en comparación con la duración total del ciclo de vida del sonido.

- *Decay* (decaimiento): Una vez que se ha alcanzado el primer máximo del sonido y, por tanto, ha concluido la región *Attack*, el sonido se desvanece hasta que alcanza una región en la que se estabiliza. La región comprendida entre el máximo y la región estable del sonido se conoce con el nombre de *Decay* y, al

igual que la región *Attack*, su duración es pequeña en comparación con la duración total del sonido.

- *Sustain* (sostenido): Es la región en la que el sonido es estable. Dicha región es la más duradera e importante de las regiones que forman el ciclo de vida. Si un músico quiere ejecutar con su instrumento una nota musical muy larga o muy corta, la parte del sonido que va a modificar es la región *Sustain*, mientras que la región de *Attack* y *Decay* permanecerán inalteradas en su duración salvo que se pretenda ejecutar un sonido que crezca más rápidamente o más lentamente, en cuyo caso se modificaría la duración de la región *Attack*.

- *Release* (liberación): Es la última región del sonido y está comprendida desde el final de la región *Sustain* hasta que se apaga completamente el sonido. La duración de esta región varía mucho dependiendo del tipo de instrumento o del tipo de ejecución. Por ejemplo, si se piensa en una nota ejecutada en un piano con una duración de 1 segundo, el sonido acaba en el momento que el músico deja de ejercer presión sobre la tecla del piano. En este caso, la región *Release* tendrá una duración muy corta respecto a la duración total del sonido, ya que comienza en el instante en que el músico suelta la tecla y acaba cuando se apaga la nota. Sin embargo, si en el mismo instrumento se pulsa una tecla y se mantiene pulsada hasta que el sonido muere de forma natural, las cuerdas del piano se van apagando progresivamente y la región *Release* tiene una duración mucho más significativa respecto a la duración total del sonido.

En la Figura 3.16 se puede apreciar un esquema de la envolvente ADSR.

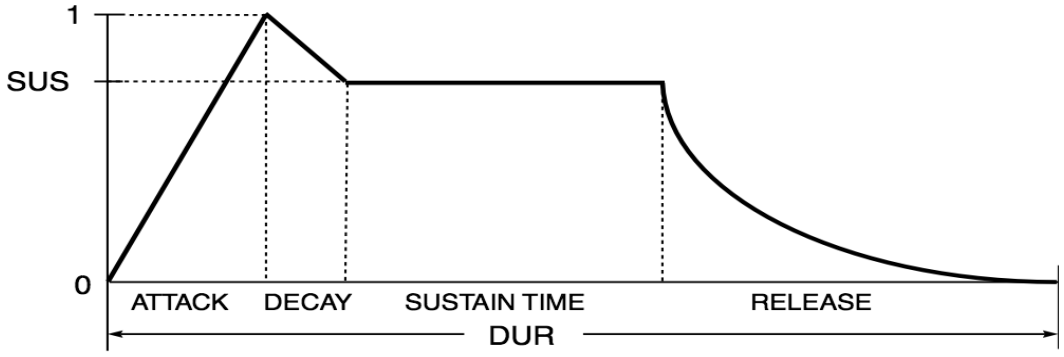


Figura 3.16. Envolvente ADSR.

El efecto *Sustain,* como su propio nombre indica, consiste en encontrar la región *Sustain* de un sonido grabado (una nota o acorde ejecutados por un instrumento musical o por una voz cantada) con el objetivo de modificar la duración de esta región y, por tanto, la duración del sonido.

Es un efecto del que no se conoce su implementación mediante técnicas basadas en las STFT.

3.7.1. PROGRAMACIÓN CWAS

Para la implementación del efecto *Sustain* mediante el algoritmo CWAS, se aplica el siguiente método:

- Primero se aplica al sonido original el algoritmo CWAS con el objetivo de obtener las matrices Módulo y Fase correspondientes.

- A continuación se busca, para cada parcial de Módulo, un tramo con una duración de 1000 *samples* cuyos valores sean estables y apenas varíen. Dicho de otra forma, se buscan tramos de 1000 *samples* que se encuentren dentro de la región *sustain* de cada parcial.

- Una vez encontrados todos los tramos estables, se almacenan los datos de su principio y fin en una matriz que se ha llamado Regiones.

- Por último, se crean dos nuevas matrices, Módulo *sustain* y Fase *sustain*, que tendrán la duración en *samples* que quiera darse al nuevo sonido. En Módulo *sustain* se repiten los tramos estables de cada parcial almacenados en Regiones tantas veces como sea necesario para alcanzar la duración del sonido deseada. En Fase *sustain* se repiten las derivadas correspondientes a las fases de los tramos estables. Realizando esta operación con las derivadas se consiguen unas transiciones mucho más suaves y se evita la aparición de *clicks* (saltos) en el sonido final.

- Finalmente se realiza la síntesis aditiva entre Módulo *sustain* y Fase *sustain,* obteniendo como resultado el sonido original alargado en su región *sustain* tanto como se desee.

3.8. VOCODER SIMPLE

El efecto llamado *Vocoder simple* es un efecto basado en lo que se conoce como *Simple Morphing*.

El *Simple Morphing* consiste en imponer las características de un sonido de control a un sonido base. Para ello se hace uso de un seguidor de envolvente que, de una forma muy simplificada, aplica la envolvente del sonido de control sobre el sonido base para que éste se ejecute siguiendo la envolvente del sonido de control.

3.8.1. PROGRAMACIÓN TRADICIONAL

La forma de implementar el efecto *Vocoder simple* mediante métodos tradicionales es la siguiente:

- Se hace uso de una señal de control y de una señal de base.

- La señal de control se eleva al cuadrado para que las variaciones en la envolvente sean más marcadas.

- Se pasa el cuadrado de la señal de control por un detector de RMS y el resultado se escala para que no exista una diferencia de volúmenes muy significativa entre la señal de control y la señal de base.

- Se multiplica el resultado anterior por la señal de base. Como resultado se obtiene la señal de base modulada por la envolvente de la señal de control.

Generalmente este efecto se aplica con una señal de control correspondiente a un fragmento de voz hablada y una señal de base que suele ser el sonido de un instrumento, ruido, etc. Tras aplicar el efecto *Vocoder simple,* el sonido producido por el instrumento sigue la envolvente del sonido hablado y las notas se ejecutan a la par de las sílabas que forman las palabras habladas.

3.8.2. PROGRAMACIÓN CWAS

Para la implementación de este efecto mediante métodos basados en el uso de la transformada Wavelet se aplica el siguiente método:

- Se ejecuta el algoritmo CWAS sobre el sonido de control y se guarda la matriz Parcial de dicho sonido (Parcial control) como un archivo externo.

- Tras este paso preliminar, se ejecuta el algoritmo CWAS sobre el sonido base para obtener la matriz Parcial base.

- Se tienen ya las matrices Parcial base y Parcial control pero, obviamente, el sonido base y el sonido control son diferentes y pueden tener distinta duración (número de filas de la matriz Parcial) y distinto número de parciales (columnas de la matriz Parcial). Por tanto, para poder trabajar con ambas matrices sin tener problemas de dimensiones es necesario adaptar una a la otra. El método usado para adaptar las matrices es el siguiente:

  - Se busca cuál de las dos matrices tiene menor número de parciales (columnas) y se le añaden tantas columnas de ceros como parciales existan de diferencia entre ambas matrices.

  - Después se comprueba cuál de las dos matrices tiene menor duración y se completa con tantas filas de ceros como *samples* de duración existan de diferencia entre ambas matrices.

- Se obtiene la matriz Módulo vocoder a partir de la matriz Parcial control por medio de la ecuación 2.8, como se ha explicado en el apartado 2.4.

- Se obtienen las matrices Módulo base y Fase base a partir de la matriz Parcial base.

- Se suman, para cada *sample* de la matriz Módulo vocoder, los componentes de cada parcial para obtener el perfil de la envolvente de volumen (ecuación 3.21).

$$Envolvente_{vocoder}(i) = \sum_{n=1}^{parciales} Módulo(i,n) \qquad \text{(ec. 3.21)}$$

$$\text{donde} \begin{cases} i = 1,...,samples \\ \\ n = 1,...,n^{o}\ parciales \end{cases}$$

- Se detecta el máximo del vector Envolvente vocoder.

- Se divide Envolvente vocoder por dicho máximo para tener el vector Envolvente vocoder normalizado entre 0 y 1.

- Se multiplica cada columna de la matriz Módulo base por el vector Envolvente vocoder normalizado (término a término) para que los módulos del sonido base sigan la envolvente del sonido de control.

- Se realiza la síntesis aditiva entre Módulo base y Fase base.

El proceso que acaba de describirse se muestra de forma esquemática en la Figura 3.17.

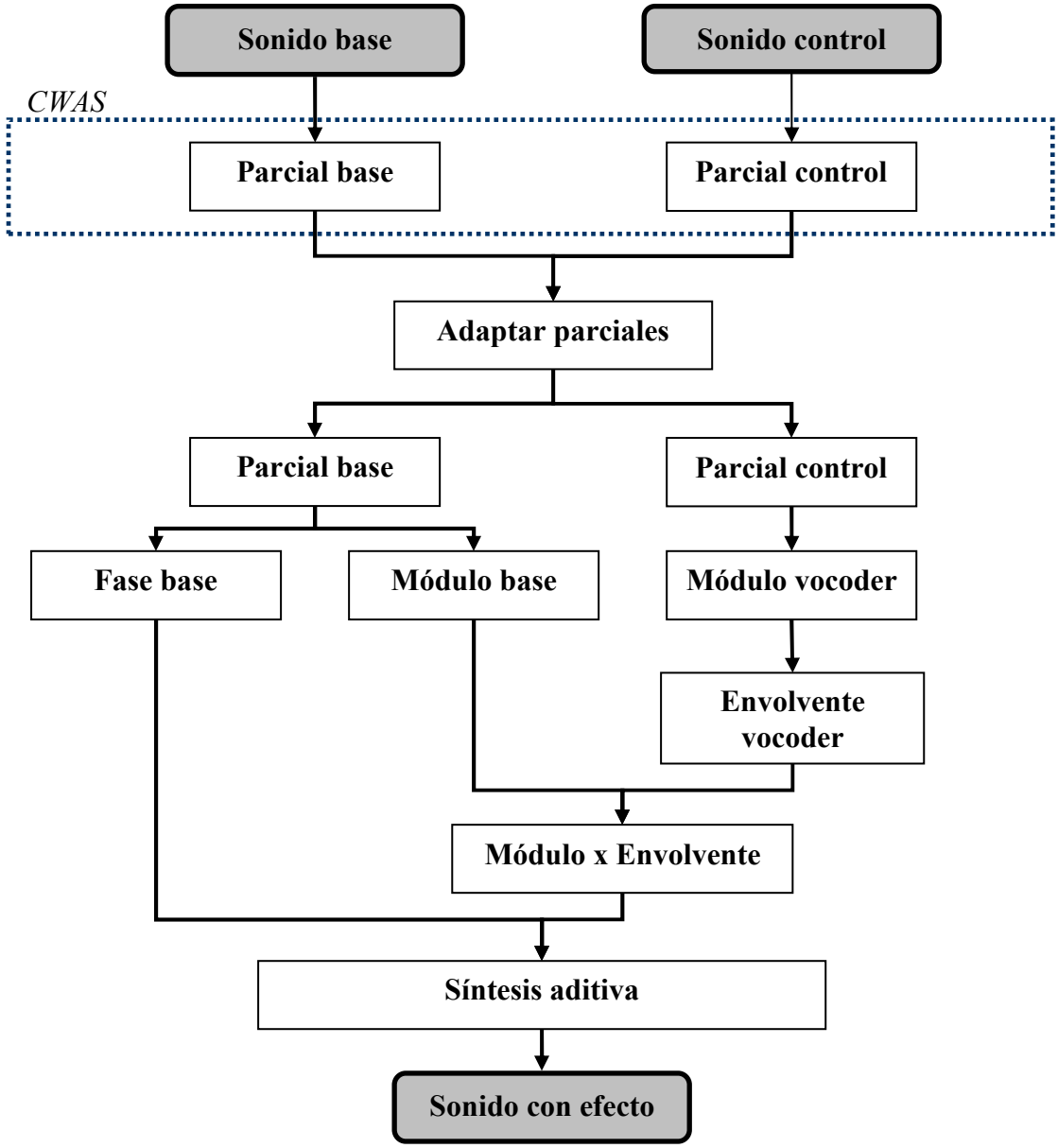


Figura 3.17. Diagrama del efecto *Vocoder simple*.

Puesto que el resultado de este efecto es un sonido base controlado por la envolvente del sonido de control se pueden conseguir combinaciones muy interesantes, como por ejemplo aplicar el vibrato de una voz a un instrumento, aplicar cualidades propias de un instrumento a otro, etc.

3.9. VOCODER MEJORADO

A diferencia del efecto anterior, el *Vocoder mejorado* es un efecto con el que se consigue que el sonido de control (generalmente una voz hablada) suene con las características de la fase del sonido base. El efecto que se produce es similar a una robotización de la voz, ya que se escuchan perfectamente las palabras pero con el tono correspondiente al sonido base.

3.9.1. PROGRAMACIÓN TRADICIONAL

Para implementar este efecto con la programación tradicional se aplican primero las STFT a cada uno de los dos sonidos para separar sus módulos y fases. Una vez hecho esto, se combinan los módulos del sonido de control con las fases del sonido base y se realiza la resíntesis del sonido, dando lugar a un sonido que es una mezcla de ambos.

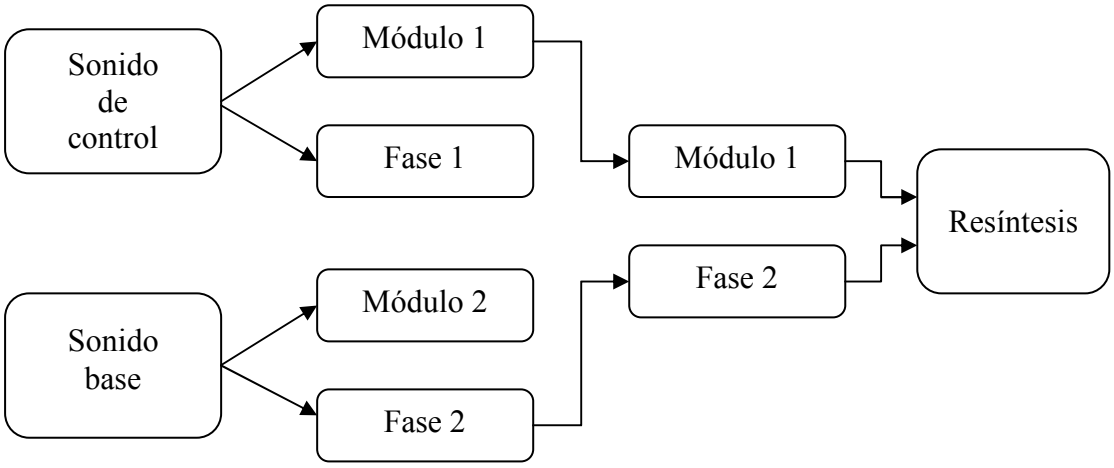


Figura 3.18. Principio de funcionamiento del efecto *Vocoder mejorado*.

3.9.2. PROGRAMACIÓN CWAS

Para la aplicación del efecto *Vocoder mejorado* desarrollado en este proyecto el sonido de control sólo puede ser una voz y el sonido de base un sonido de cualquier instrumento musical. El efecto que se consigue es similar a la robotización, ya que la voz suena con el tono del instrumento. Si el sonido del instrumento es una única nota musical, el efecto *Vocoder mejorado* es acústicamente muy similar al efecto de *Robotización.*

El esquema de la implementación del efecto *Vocoder mejorado* a partir del algoritmo CWAS puede apreciarse en la Figura 3.19.

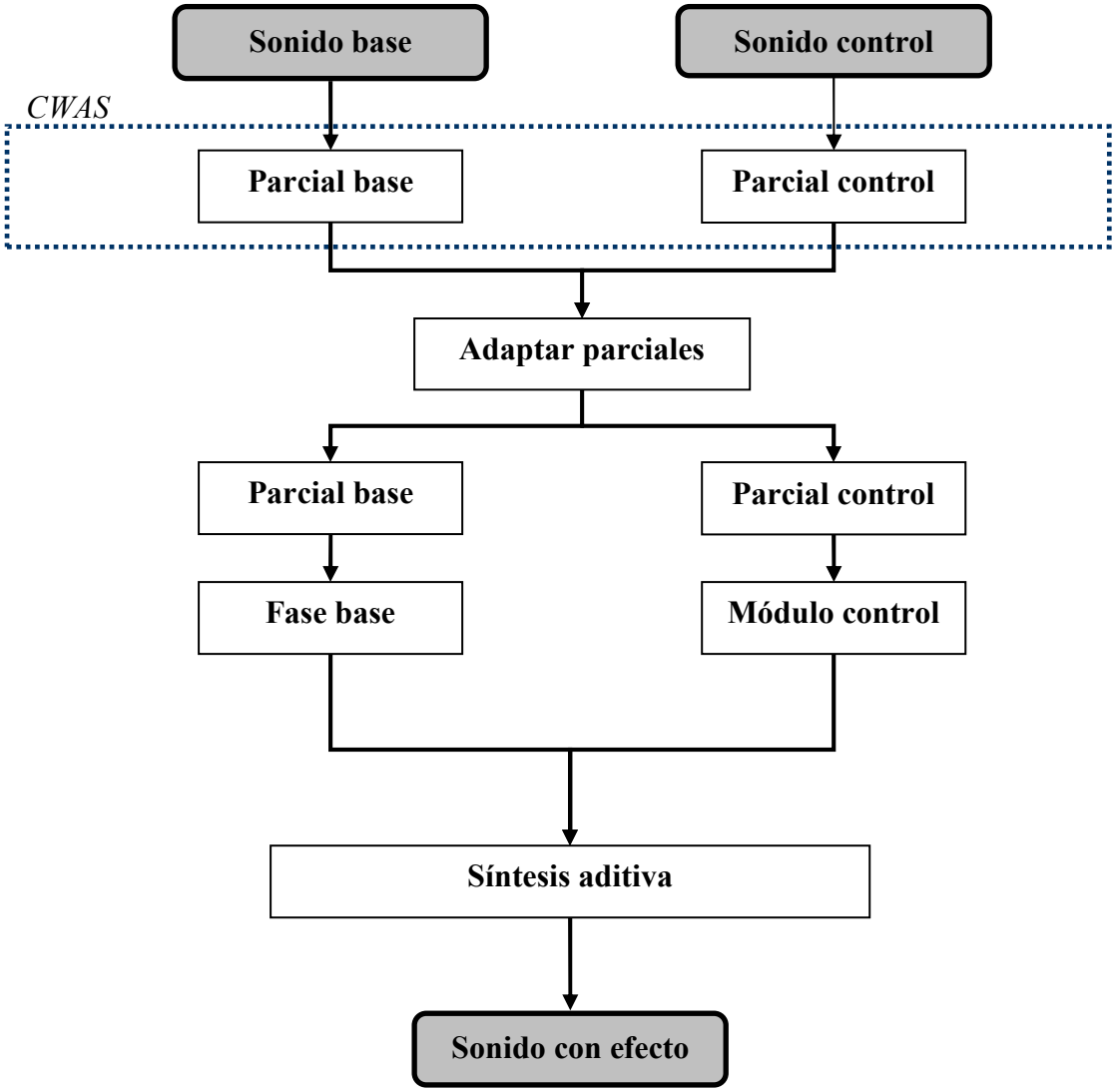


Figura 3.19. Diagrama del efecto *Vocoder mejorado*.

Como se puede apreciar en la Figura 3.19, se trata de un efecto bastante sencillo de implementar:

- Se ejecuta el algoritmo CWAS sobre sonido de control y se guarda su matriz Parcial (Parcial control).

- Se aplica la CWAS al sonido de base y se obtiene su matriz Parcial (Parcial base).

- Se adaptan ambas matrices (Parcial base y Parcial control) de manera que el número de filas sea el número de *samples* de duración de la señal más corta y el número de columnas sea el correspondiente a la matriz con mayor número de parciales.

- Se obtienen Módulo control y Fase base.

- Se realiza la síntesis aditiva entre Módulo control y Fase base para obtener el sonido de salida.

3.10. MORPHING

El efecto llamado *Morphing* es un efecto que implica la transformación gradual de un sonido en otro a través de la combinación de las cualidades de ambos sonidos. A diferencia del *Vocoder mejorado*, en el que se combina el módulo de un sonido con la fase de otro, con el efecto *Morphing* se transfieren las características de un sonido a otro. El *Morphing* se puede usar, por tanto, para generar sonidos híbridos, sonidos con características de dos instrumentos distintos y sonidos que evolucionan de un sonido a otro.

Hay que destacar que este efecto conseguirá un resultado más realista cuando se utilicen dos sonidos que difieran en tono lo mínimo posible ya que, de lo contrario, se producirá una variación muy brusca de un sonido a otro y el resultado será poco natural.

Es un efecto del que no se conoce su implementación por los métodos tradicionales.

3.10.1. PROGRAMACIÓN CWAS

Para implementar el efecto *Morphing* mediante el algoritmo CWAS deben obtenerse los módulos y fases de los distintos parciales que forman los dos sonidos que quieren combinarse y, a continuación, mezclarlos progresivamente para transformar así los parciales de un sonido en los parciales del otro. El usuario puede elegir los porcentajes de sonido A y de sonido B que desea que formen parte del sonido final.

A continuación se detallan los pasos necesarios para implementar el efecto *Morphing* mediante el algoritmo CWAS:

- Se ejecuta el algoritmo CWAS sobre los sonidos A y B y se obtienen sus matrices Parcial A y Parcial B respectivamente.

- Se adaptan las matrices Parcial A y Parcial B de manera que las matrices resultantes tengan el mayor número de parciales de entre A y B y la longitud en *samples* del sonido más corto. Cuando sea necesario aumentar alguna de las matrices originales (Parcial A o Parcial B iniciales), se completa hasta el nuevo tamaño con ceros.

- Se calculan las matrices Módulo A, Módulo B, Fase A y Fase B.

- Se combinan módulos y fases de acuerdo con la ecuación 3.22. Las nuevas matrices de módulos y fases que se obtienen reciben el nombre de Módulo morphing y Fase morphing respectivamente.

$$Módulo_{morphing} = m \times Módulo_B + (1-m) \times Módulo_A$$
$$Fase_{morphing} = m \times Fase_B + (1-m) \times Fase_A \quad \text{(ec. 3.22)}$$

  donde el parámetro *m* (parámetro *mezcla* en el *script* de *Matlab*) determina el porcentaje del sonido B que estará presente en el sonido final.

- Por último se realiza la síntesis aditiva entre Módulo morphing y Fase morphing.

El procedimiento que acaba de describirse se muestra de forma esquemática en la Figura 3.20.

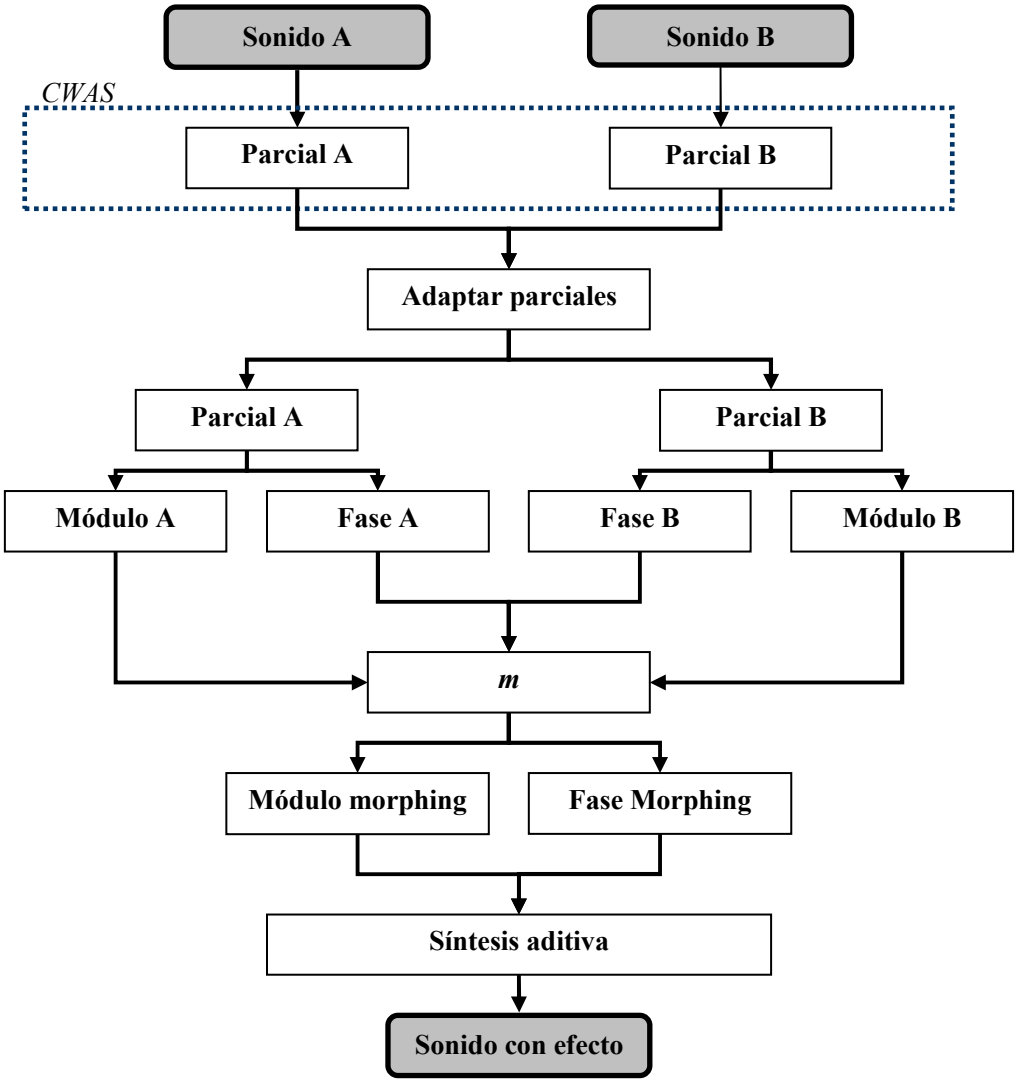


Figura 3.20. Diagrama del efecto *Morphing*

3.11. PSEUDOROBOT

El efecto *Pseudorobot* nace de un intento de adaptar el algoritmo del efecto *Robotización* mediante STFT a su uso con la CWAS. No se consigue exactamente una robotización ya que, al comparar las formas de onda del efecto *Pseudorobot* con las formas de onda de una *Robotización* tradicional, se observa que son muy diferentes. Sin embargo, a efectos sonoros se consigue un efecto bastante similar.

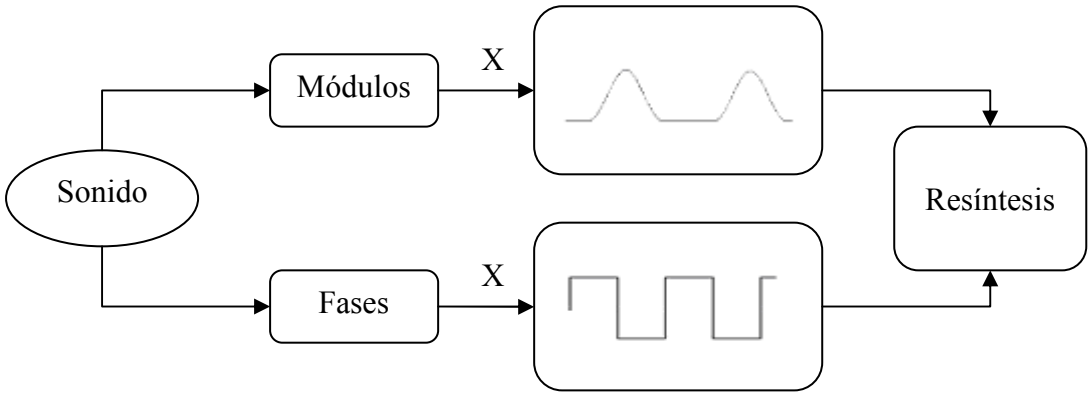


Figura 3.21. Principio de funcionamiento del efecto *Pseudorobot*.

3.11.1. PROGRAMACIÓN CWAS

La implementación CWAS del efecto *Pseudorobot* consiste en lo siguiente:

- Se aplica el algoritmo CWAS al sonido original y se obtienen sus matrices Módulo y Fase.
- Se genera una onda cuadrada, que oscilará entre 0 y 1, cuya extensión total sea igual al número de *samples* totales del sonido,
- Se multiplica cada columna de Fase (cada parcial) por la onda cuadrada.
- Se multiplica cada columna de Módulo por una onda cuadrada suavizada. Con los módulos es necesario trabajar con una onda cuadrada suavizada para que las transiciones entre 0 y 1 no sean tan bruscas, evitándose así ruidos no deseados. Este suavizado se consigue multiplicando cada tramo de la onda cuadrada con valor 1 por una ventana de *hanning* del mismo tamaño.

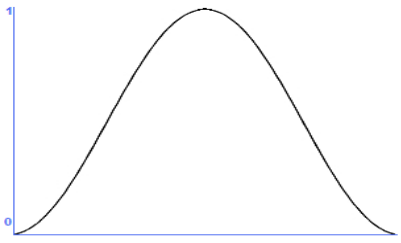


Figura 3.22. Ventana de *Hanning*.

- Por último se realiza la síntesis aditiva entre las nuevas matrices Módulo y Fase.

Después de realizar varias pruebas de sonido con este efecto se llegó a la conclusión de que el tamaño adecuado para la onda cuadrada era de 167 *samples* por semiperiodo (334 *samples* por periodo), tamaño que se corresponde con una frecuencia de onda de:

para f_s=22050 Hz $\begin{cases} T = \dfrac{2 \times 167}{f_s} = 15{,}15ms \\ f = \dfrac{1}{T} = 66Hz \end{cases}$

3.12. ARMONIZADOR

El efecto *Armonizador* es un efecto que consiste en desplazar el sonido original varios semitonos (como se hace en el *Pitch-shifting*) y generar un nuevo sonido formado por la suma del sonido original y el sonido desplazado, de manera que se crean melodías paralelas.

A continuación, para entender un poco mejor la base de este efecto, se introduce una breve explicación sobre acordes e intervalos.

Un intervalo es la distancia en semitonos que existe entre dos notas distintas. Por su parte, se llama acorde musical a la ejecución simultánea de varias notas.

Los acordes más típicos son los acordes mayores y los acordes menores. Los acordes mayores están formados por la nota base o fundamental más un intervalo de tercera mayor (4 semitonos) y un intervalo de quinta justa (7 semitonos) sobre la nota fundamental del acorde. Por ejemplo, un acorde de Do mayor está formado por las notas Do, Mi y Sol. Por otro lado los acordes menores están formados por la nota fundamental más un intervalo de tercera menor (3 semitonos) y un intervalo de quinta justa (7 semitonos) sobre la nota fundamental. Por ejemplo, un acorde de Do menor está formado por las notas Do, Mi bemol y Sol.

Ambos tipos de acordes (mayores y menores) contienen una quinta justa (7 semitonos) sobre la nota fundamental, de manera que, independientemente de que se pretenda ejecutar un acorde mayor o menor, si se añade una nota desplazada 7 semitonos por encima de la nota base se creará un efecto con un sonido agradable. Es por ello que para implementar el efecto *Armonizador* se ha elegido desplazar el sonido base 7 semitonos.

Al tratarse de un efecto desarrollado por primera vez en este proyecto, no se conoce su implementación mediante los métodos de programación tradicionales.

3.12.1. PROGRAMACIÓN CWAS

El efecto *Armonizador* está directamente relacionado con el *Pitch-shifting simple* y, de hecho, el *Pitch-shifting* se puede considerar como un bloque propio del diagrama de bloques del efecto Armonizador (Figura 3.23).

La implementación CWAS del efecto *Armonizador* consiste en los siguientes pasos:

- Se aplica el algoritmo CWAS al sonido original y se obtienen las matrices Módulo y Fase.

- Se duplican las matrices Módulo y Fase para poder trabajar con dos señales (Módulo 1 y Fase 1; Módulo 2 y Fase 2). Las matrices Módulo 1 y Fase 1 se mantendrán inalteradas para hacer la resíntesis del sonido original. Por otro lado, a las matices Módulo 2 y Fase 2 se les aplicará el *Pitch-shifting*.

- Se multiplica Módulo 2 por el parámetro υ (parámetro *volumen* en el *script* de *Matlab* correspondiente), que determina el volumen del sonido duplicado respecto del sonido original. El resultado es la matriz Módulo suma.

- Se aplica un *Pitch-Shifting* de 7 semitonos por encima de la fase original a Fase 2 (mismo procedimiento que en el apartado 3.5.2). El resultado es la matriz Fase suma.

- Se realiza la síntesis aditiva entre Módulo 1 y Fase 1 por un lado y Módulo suma y Fase suma por otro, obteniéndose el sonido original sintético y el sonido a sumar, respectivamente.

- Se suman ambos sonidos y se obtiene el sonido de salida del efecto.

En la Figura 3.23 se muestra un esquema del procedimiento que acaba de describirse.

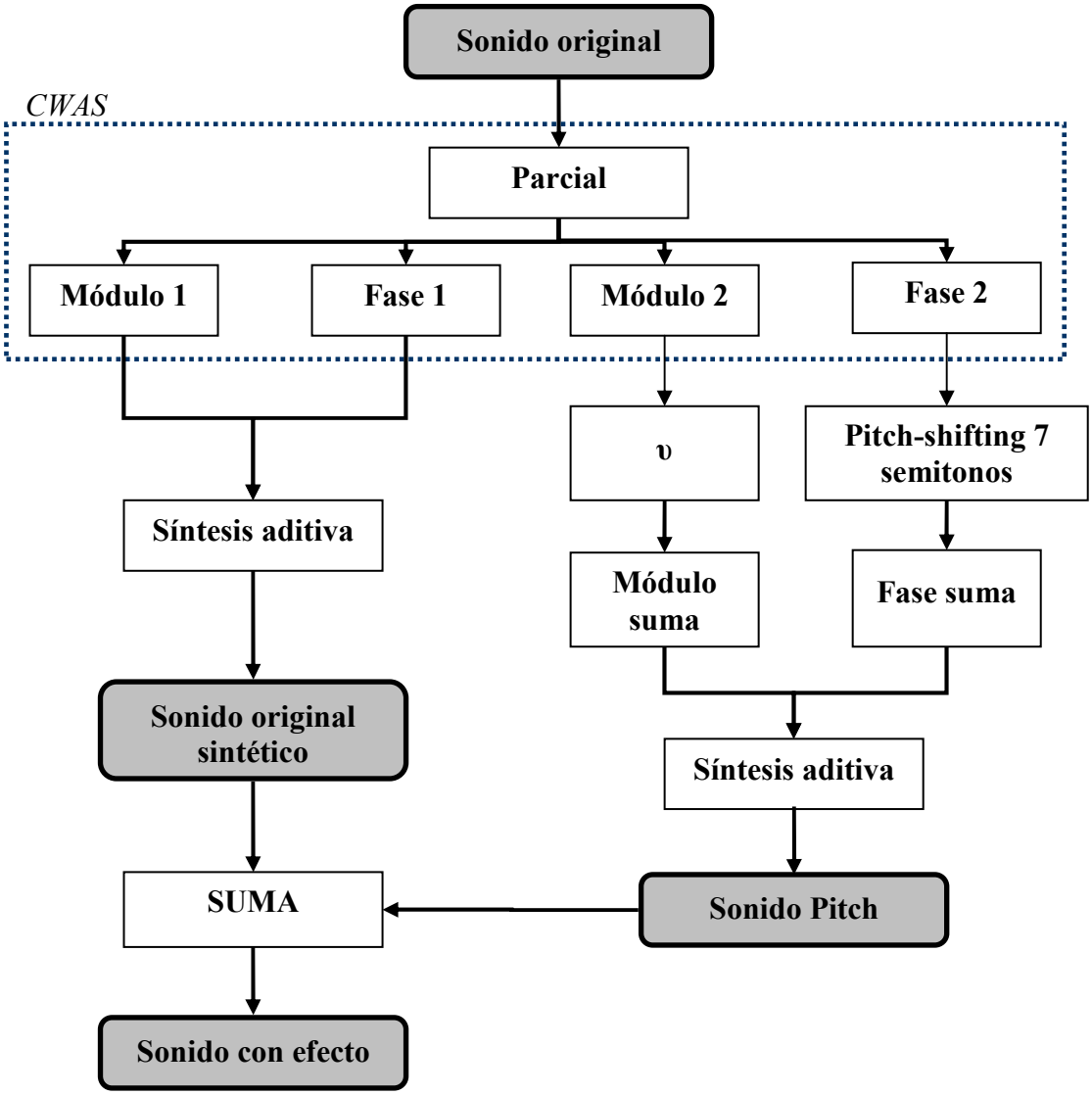


Figura 3.23. Diagrama del efecto *Armonizador*.

3.13. REVERB

La reverberación es un fenómeno acústico natural que se produce en recintos más o menos cerrados, por el cual a la señal original se le van sumando las diferentes ondas reflejadas en las paredes del recinto con un retardo o *delay*, generado básicamente por la distancia física entre la fuente de sonido original y las paredes del recinto.

El oído humano es incapaz de distinguir un sonido de sus reflexiones cuando la diferencia de tiempo entre ambas es menor de 66 ms (persistencia acústica). Además, según el efecto HAAS (conocido también como efecto de precedencia), cuando el oído capta unas reflexiones con un retardo no superior a 35 ms es incapaz de integrarlas como ecos consecutivos y las asocia a la fuente de sonido original, reforzándola en intensidad.

A la hora de grabar sonidos musicales es interesante que el sonido grabado se escuche de la misma forma que el sonido en vivo y, para ello, es necesario el uso del efecto *Reverb,* que consiste en recrear las condiciones acústicas de distintos recintos cerrados (habitación, sala de conciertos, iglesia,…). Puesto que existen multitud de recintos cerrados, cada uno con distintas propiedades acústicas, se pueden definir también distintos tipos del efecto *Reverb* [6]. Algunos de los más importantes se citan a continuación:

- *Hall:* Simula una gran sala, similar a un auditorio.
- *Room:* Simula las condiciones acústicas de una sala pequeña o una habitación.
- *Plate:* Realmente es una reverberación mecánica, como si se colocase una plancha metálica frente a la fuente de sonido, de forma que las reflexiones primarias son muy brillantes.
- *Cathedral:* Intenta recrear las condiciones acústicas de la típica catedral gótica. Es una reverberación muy larga, densa y llena de matices, pero poco recomendable para las mezclas ya que tiende a ensuciar demasiado el sonido.
- *Teathre*: En teoría simula las condiciones acústicas de un teatro, aunque en la práctica viene a ser como un efecto *Reverb Hall* un poco menos denso y con menos reflexiones primarias.
- *Studio:* En principio, un estudio de grabación no debería tener ningún tipo de reverberación, pero un efecto *Studio Reverb* es una reverberación en la que se mantienen las reflexiones primarias y se eliminan el resto.

3.13.1. PROGRAMACIÓN TRADICIONAL

La *Reverb* que se ha tomado como referencia para su desarrollo en el presente proyecto es una *Reverb* básica, del tipo *Reverb Hall*. Su implementación mediante métodos tradicionales es sencilla y se corresponde con la denominada "*Recursive Delay Line*" (ecuación 3.23).

$$y(n) = -g \cdot x(n) + x(n - delay) + g \cdot y(n - delay) \qquad \text{(ec. 3.23)}$$

donde *y(n)* es la señal de salida para el *sample* n, *x(n)* es la señal de entrada para el *sample* n, *delay* es la longitud en *samples* del tiempo de retraso y *g* es el factor de atenuación que sufre la señal.

3.13.2. PROGRAMACIÓN CWAS

La implementación del efecto *Reverb Hall* con el algoritmo CWAS no puede realizarse directamente con la ecuación de la "*Recursive delay line"* ya que al aplicar la CWAS se trabaja con matrices de un tamaño considerable, lo que supondría un tiempo de cálculo muy elevado. Para evitar este problema, la reverberación de una sala se ha planteado como la suma de la señal fuente más las señales reflejadas en cada pared del recinto cerrado.

Como se aprecia en la Figura 3.24, se ha elegido una sala de forma cuadrada en la que el usuario del efecto puede elegir las siguientes dimensiones:

- *dim*: con este parámetro se establece el ancho, en metros, de la sala cuadrada. La sala tendrá unas dimensiones de *dim*×*dim* metros cuadrados.

- *x* e *y*: establecen las distancias respecto a los ejes de coordenadas a las que se encuentra la fuente del sonido.

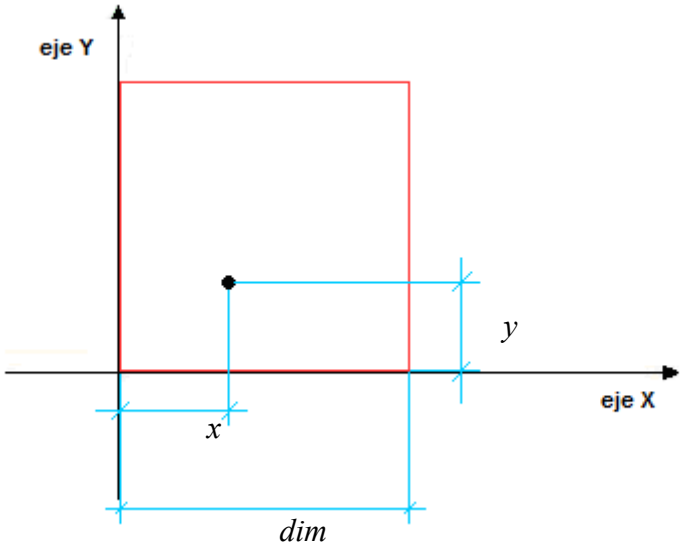


Figura 3.24. Dimensiones y forma de la sala empleada para la *Reverb Hall*.

Para implementar el efecto *Reverb Hall* se han tenido en cuenta los siguientes supuestos:

- El sonido que se percibirá en el punto en el que se encuentra la fuente será la suma del sonido fuente más el sonido reflejado en cada una de las cuatro paredes. La reflexión en las paredes es, en realidad, una suma de muchas reflexiones ya que el sonido se propaga de forma esférica. Sin embargo, para simplificar el algoritmo y reducir el tiempo de cálculo, se ha considerado que el sonido se refleja siguiendo una línea recta y perpendicular a cada pared.

- La longitud que recorre el sonido reflejado es el doble de la distancia existente entre la fuente y la pared.

- Se ha considerado que cuando un sonido es reflejado queda atenuado por un factor *g*, que está directamente relacionado con la longitud que recorre el sonido. El factor *g* elegido viene dado por la ecuación 3.24.

$$g = 1 - \frac{longitud}{100} \qquad \text{(ec. 3.24)}$$

- La velocidad del sonido se considera v = 340 m/s.

- Se considera que cada reflexión conlleva un retraso respecto al sonido fuente, de manera que ese tiempo de *delay* está relacionado con la velocidad del sonido y con la longitud que recorre según la ecuación 3.25.

$$t_{delay}(s) = \frac{longitud(m)}{v(m/s)} \qquad \text{(ec. 3.25)}$$

- A su vez, el número de *samples* de *delay* para cada reflexión viene dado por la ecuación 3.26.

$$samples_{delay} = t_{delay}(s) \times f_s(Hz) \qquad \text{(ec. 3.26)}$$

donde f_s es la frecuencia de muestreo.

Teniendo en cuenta todas estas consideraciones, la implementación CWAS del efecto *Reverb* consiste en los siguientes pasos:

- Se calcula el vector lon, que tiene cuatro casillas en las que se almacenan las longitudes que recorre cada onda reflejada.

- Se obtiene el vector G, que contiene los 4 factores de atenuación que se aplican a cada onda reflejada.

- Se obtiene el vector T, que contiene los tiempos de *delay* de cada reflexión.

- Se obtiene el vector S, que contiene los *samples* de *delay* asociados a cada reflexión.

- Se obtiene el máximo del vector S y se guarda en la variable *maxi*.

- Se aplica el algoritmo CWAS al sonido original y se obtienen las matrices Parcial, Módulo y Fase.

- Se redimensionan las matrices Módulo y Fase añadiendo ceros a cada columna asociada a cada parcial hasta completar la duración máxima que puede tener la señal final. Esta duración máxima se corresponde con la longitud original (en *samples*) más el máximo número de *samples* de *delay* (almacenado en *maxi*).

- Se generan cuatro matrices Módulo y cuatro matrices Fase, una por cada reflexión. Al inicio de cada una de estas matrices se aplica un *zero-padding* según los valores almacenados en S ($samples_{delay}$) y al final de cada una se aplica otro *zero-padding* de (*maxi* - $samples_{delay}$) samples.

- Cada matriz módulo se multiplica por su factor de atenuación correspondiente, almacenado en G.

- Se realiza la síntesis aditiva entre las matrices Módulo y Fase correspondientes a cada una de las reflexiones, obteniendo cuatro señales de audio.

- Se realiza la síntesis aditiva entre las matrices Módulo y Fase originales (una vez readaptadas a la duración máxima) y se obtiene el sonido fuente.

- El sonido de salida del efecto se obtiene sumando las cinco señales de audio correspondientes al sonido fuente y a los cuatro sonidos reflejados en las paredes.

Aunque no se ha realizado en este proyecto, el tono de cada reflexión podría modificarse fácilmente, de manera que, además de introducir un *delay* en cada reflexión, cambiaría ligeramente su frecuencia. El resultado sería un nuevo efecto *Reverb,* un poco alejado del concepto tradicional de reverberación, pero que podría resultar interesante para algunas aplicaciones.

3.14. TRÉMOLO

El *Trémolo* es un efecto similar al *Vibrato*. Tanto es así, que a veces ambos términos se confunden, pero existe una diferencia entre ellos. Mientras que el *Vibrato* es una leve oscilación de la frecuencia de un sonido, el *Trémolo*, por su parte, consiste en una leve oscilación sinusoidal de la amplitud del sonido manteniendo su frecuencia constante.

3.14.1. PROGRAMACIÓN TRADICIONAL

La forma de implementar el *Trémolo* mediante métodos tradicionales es muy sencilla y consiste en multiplicar la totalidad de la onda sonora por una señal sinusoidal de baja frecuencia (entre 1 y 10 Hz). Si se utiliza una onda sinusoidal mayor de 10 Hz para modular, el sonido se desnaturaliza y se produce un cambio en el tono original de la señal.

3.14.2. PROGRAMACIÓN CWAS

El procedimiento seguido para implementar el efecto *Trémolo* mediante el algoritmo CWAS es el siguiente:

- Se aplica el algoritmo CWAS al sonido original y se obtienen las matrices Módulo y Fase.

- Se genera una onda sinusoidal, que oscile entre 0 y 1 y que tenga la misma extensión temporal que el sonido y una frecuencia entre 1 y 10 Hz, a elegir por el usuario. Para ello se aplica la ecuación 3.27.

$$y = A \cdot \sin(2\pi \cdot f \cdot t) + 0.5 \qquad \text{(ec. 3.27)}$$

  donde A=0,5 para que la onda sinusoidal oscile entre 0 y 1.

  No se puede usar una onda sinusoidal normal que oscile entre -1 y 1 ya que los valores de los módulos que se obtienen de la CWAS son siempre positivos y se podría generar un conflicto al multiplicarlos por valores negativos.

- Se multiplica cada columna de la matriz Módulo (cada parcial) por la onda sinusoidal generada.

- Se realiza la síntesis aditiva entre las matrices Módulo (tras haberla multiplicado por la onda sinusoidal) y Fase, obteniéndose el sonido de salida con el *Trémolo* aplicado.

3.15. OVERDRIVE Y DISTORSIÓN CON SIMULACIÓN VALVULAR

El efecto *Overdrive* trata de imitar el recorte de la señal que se produce de forma natural en los amplificadores de válvulas cuando éstos se saturan. Se puede decir que el efecto *Overdrive* consiste en amplificar, en una primera etapa, la señal de entrada para, de forma suave, llevar esa amplificación a una región no lineal. Dicho en otras palabras, por encima de cierto valor de la señal de entrada se produce una ligera distorsión de la misma, mientras que para los valores más bajos se produce una amplificación lineal de la señal.

Por otro lado, el efecto *Distorsión* permite amplificar ligeramente la señal de entrada y llevarla, en su mayor parte, a la región no lineal, distorsionando prácticamente la totalidad de la onda original.

Una de las diferencias entre *Overdrive* y *Distorsión* es su aplicación en los distintos tipos de música. El *Overdrive* es un efecto que se usa más en tipos de música como el blues y el pop, mientras que el efecto *Distorsión* se utiliza en el rock o en estilos más pesados como el metal.

En la implementación de los efectos *Overdrive y Distorsión* realizada en este proyecto se ha intentado avanzar un paso más, incluyendo una simulación de la respuesta que tienen dichos efectos cuando son producidos por un amplificador de válvulas, con el objetivo de obtener un resultado más musical y agradable.

Aunque tecnológicamente las válvulas de vacío son componentes electrónicos obsoletos, siguen teniendo un gran uso en la fabricación de amplificadores de guitarras y bajos eléctricos y en los equipos de Hi-Fi. A continuación se va a introducir una breve reseña sobre el uso de los amplificadores de válvulas, que ayudará a entender por qué la mayor parte de los músicos actuales siguen usando este tipo de equipos a pesar de estar tecnológicamente obsoletos.

Las válvulas fueron los dispositivos electrónicos activos por excelencia desde principios del siglo XX hasta que en los años sesenta fueron sustituidos por los transistores y diodos de estado sólido, capaces de desempeñar las mismas funciones en un espacio mucho más reducido, con un menor peso y con una temperatura de funcionamiento muy inferior a la de las válvulas. A principios de los años setenta empezaron a surgir nuevas empresas que apostaban por la amplificación a transistores, mientras que las empresas ya consolidadas ampliaban sus catálogos con este tipo de amplificadores para no perder ventas ni mercado. Pero desde sus inicios el transistor mostró con grave problema: su linealidad y su mejor rendimiento teórico daban como resultado un sonido muy frío y con poco carácter en circuitos de audio. Por esta razón la válvula se ha mantenido desde entonces en amplificadores para instrumentos musicales y para aplicaciones de audio profesionales. Su comportamiento no lineal y “teóricamente imperfecto” da lugar a sonidos mucho más musicales y atractivos en cuanto a su tonalidad. Ni siquiera un complejo circuito digital es capaz de emular totalmente el comportamiento de una válvula.

Figura 3.25. Válvula de vacío modelo 12ax7 (doble triodo).

La saturación natural de un amplificador de válvulas se consigue, básicamente, recortando la señal de entrada. Cuanto más se recorta la señal, es decir, cuanto más se asemeja a una onda cuadrada, más se satura la señal y más se distorsiona su forma de onda con respecto a la forma de onda original.

En los amplificadores valvulares esta distorsión se consigue utilizando varias etapas de amplificación con grandes ganancias, haciendo que las válvulas lleguen a saturarse y recorten así la señal de entrada. Este tipo de distorsión natural genera armónicos impares, aumentando la presencia de los armónicos 3 y 5 del sonido. Además de incrementar los armónicos 3 y 5, cuando se satura con un amplificador de válvulas se realza también de forma natural el armónico 2 de los sonidos [7].

El oído humano percibe como más musicales los armónicos impares. Ésta es la razón por la que una distorsión procedente de un circuito a válvulas resulta mucho más musical y atractiva.

3.15.1. PROGRAMACIÓN TRADICIONAL

Uno de los métodos empleados para simular el efecto *Overdrive* es el conocido como "*Symmetrical Soft Clipping*", cuya aplicación sobre una señal normalizada viene dada por la ecuación 3.28.

$$f(x) = \begin{cases} 2x & \text{para } 0 \le x \le 1/3 \\ \dfrac{3-(2-3x)^2}{3} & \text{para } 1/3 \le x \le 2/3 \\ 1 & \text{para } 2/3 \le x \le 1 \end{cases} \qquad \text{(ec. 3.28)}$$

Este método se aplica sobre la señal de entrada en su totalidad, sin diferenciar módulos de fases.

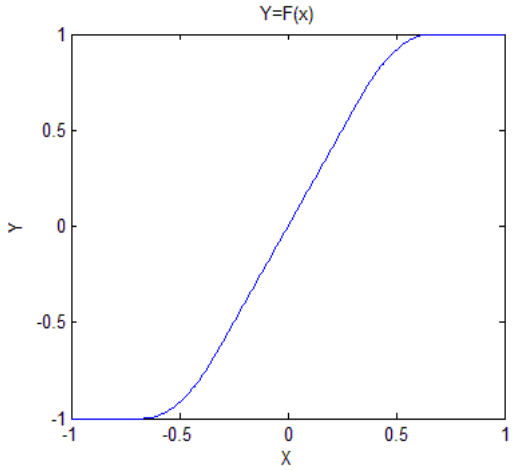


Figura 3.26. Función del *Overdrive.*

Por otro lado, para producir el efecto *Distorsión* se ha utilizado la función no lineal que se muestra en la ecuación 3.29. Esta función se aplica sobre la totalidad de la señal de audio, sin diferenciar entre módulos y fases.

$$f(x) = \frac{x}{|x|}\left(1 - e^{x^2/|x|}\right) \qquad \text{(ec. 3.29)}$$

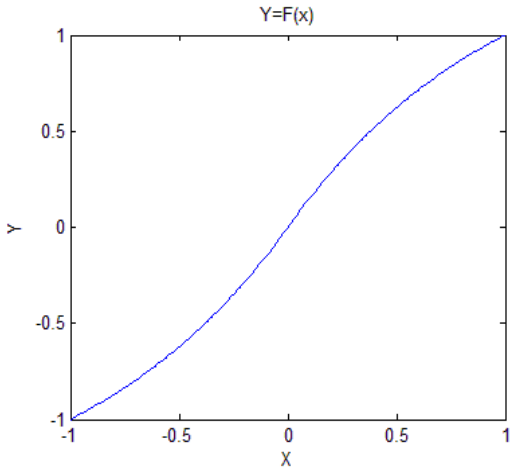


Figura 3.27. Función no lineal de la *Distorsión*.

3.15.2. PROGRAMACIÓN CWAS

A los efectos típicos de *Overdrive* y *Distorsión* descritos anteriormente se les ha aplicado un algoritmo basado en el uso de la CWAS, que consiste en encontrar los armónicos que normalmente se realzan en los amplificadores de válvulas de vacío (armónicos 2, 3 y 5). El objetivo es que el sonido se parezca más al que se consigue cuando se utilizan dichos amplificadores para generar un efecto *Overdrive* o una *Distorsión*. La manera de realzar estos armónicos se explica esquemáticamente en la Figura 3.28.

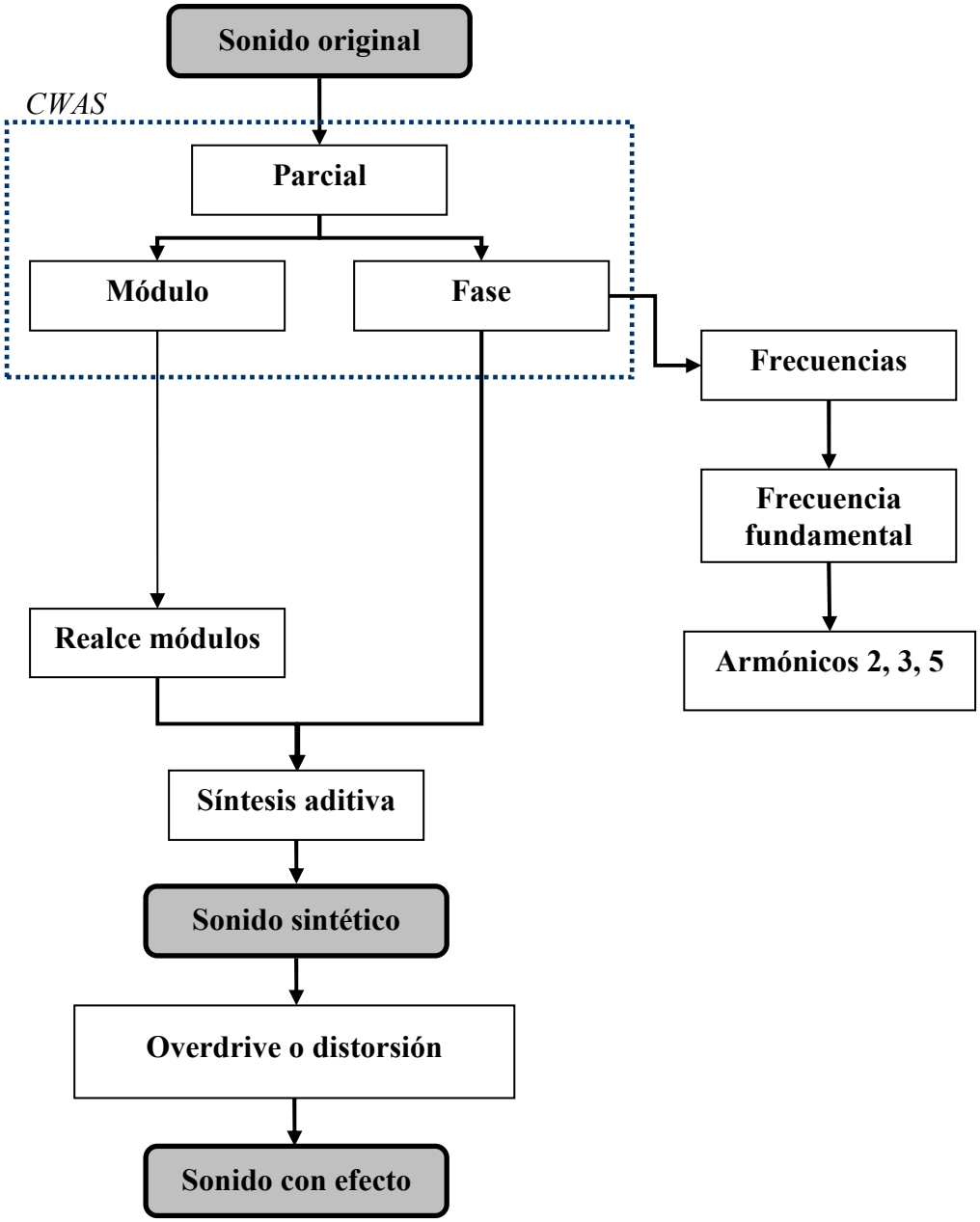


Figura 3.28. Diagrama de los efectos *Overdrive* y *Distorsión* con simulación valvular.

El proceso seguido para implementar los efectos *Overdrive* y *Distorsión* con simulación valvular es el siguiente:

- Se aplica el algoritmo CWAS al sonido original y se obtienen sus matrices Módulo y Fase.
- A partir de la matriz Fase se calculan las frecuencias instantáneas con la ecuación 3.17.
- A partir de las frecuencias instantáneas se localiza la frecuencia fundamental del sonido.

- Una vez localizada la frecuencia fundamental, se obtienen las frecuencias de los armónicos 2, 3 y 5. Para ello se utiliza la ecuación 3.30.

$$f_{armónico} = f_{fundamental} \times n^{o}\, armónico \qquad \text{(ec. 3.30)}$$

- A continuación, para saber qué parciales se corresponden con los armónicos 2, 3 y 5, se buscan las frecuencias que más se aproximan a las de dichos armónicos dentro de las frecuencias medias de cada parcial.

- Una vez se conocen los parciales correspondientes a cada armónico, se realzan dichos armónicos multiplicando los módulos de esos parciales por 3.

- Se realiza la síntesis aditiva entre la matriz Módulo (con los armónicos realzados) y la matriz Fase.

- Se aplica una de las funciones correspondientes a los efectos *Overdrive* o *Distorsión* sobre el sonido sintético, obteniéndose el sonido de salida con el efecto aplicado. Para aplicar el efecto *Overdrive* se multiplica la señal sintética por la función correspondiente al "*Simmetrical Soft Clipping*", función que se muestra en la ecuación 3.28. Para aplicar el efecto *Distorsión* se multiplica la señal sintética por la función no lineal mostrada en la ecuación 3.29.

4. RESULTADOS

En este apartado de la memoria se analizan los resultados obtenidos al implementar los efectos musicales a partir del algoritmo CWAS. Los resultados obtenidos se han comparado con aquéllos que se obtienen cuando se utilizan métodos tradicionales para la implementación de los mismos efectos.

4.1. SLAPBACK-ECHO

En el efecto *Slapback-Echo* implementado en este proyecto a partir del algoritmo CWAS se han introducido algunas variaciones con respecto al efecto tradicional que consisten en lo siguiente:

- Posibilidad de cambiar el volumen con el que se ejecuta la repetición del sonido a través de un parámetro ajustable por el usuario (parámetro *g*).

- Posibilidad de modificar el tono del sonido repetido, de manera que el usuario puede elegir entre una repetición pura de la señal de audio original o una repetición modificada cambiando el parámetro *tn*.

A continuación se muestran los resultados obtenidos tras la aplicación de los efectos *Slapback-Echo* tradicional y *Slapback-Echo* CWAS a dos señales de audio distintas: una voz humana y un sonido de piano.

Voz Humana

En este caso se ha aplicado el efecto *Slapback-Echo* configurado para funcionar como *Echo* (tiempo de *delay* de 100 ms).

La Figura 4.1(a) muestra la forma de onda de la señal original, en este caso una palabra pronunciada por una voz humana masculina.

En las Figuras 4.1(b) y 4.1(c) se observan las formas de onda de la señal tras aplicar el efecto *Slapback-Echo* tradicional (señal original más señal repetida) y de la señal repetida, respectivamente. Por otro lado, las Figuras 4.1(d) y 4.1(e) muestran las mismas formas de onda pero, esta vez, tras aplicar el efecto *Slapback-Echo* CWAS.

El *Slapback-Echo* CWAS produce una modificación más pronunciada de los parámetros de la señal repetida por lo que, al sumar dicha señal a la señal original, se obtiene una forma de onda ligeramente distinta a la obtenida con el efecto tradicional.

Al aplicar el efecto *Slapback-Echo* CWAS se ha introducido una ligera modificación del tono de la señal (*tn*=1,01), pero dicha modificación no puede apreciarse analizando sólo las formas de onda.

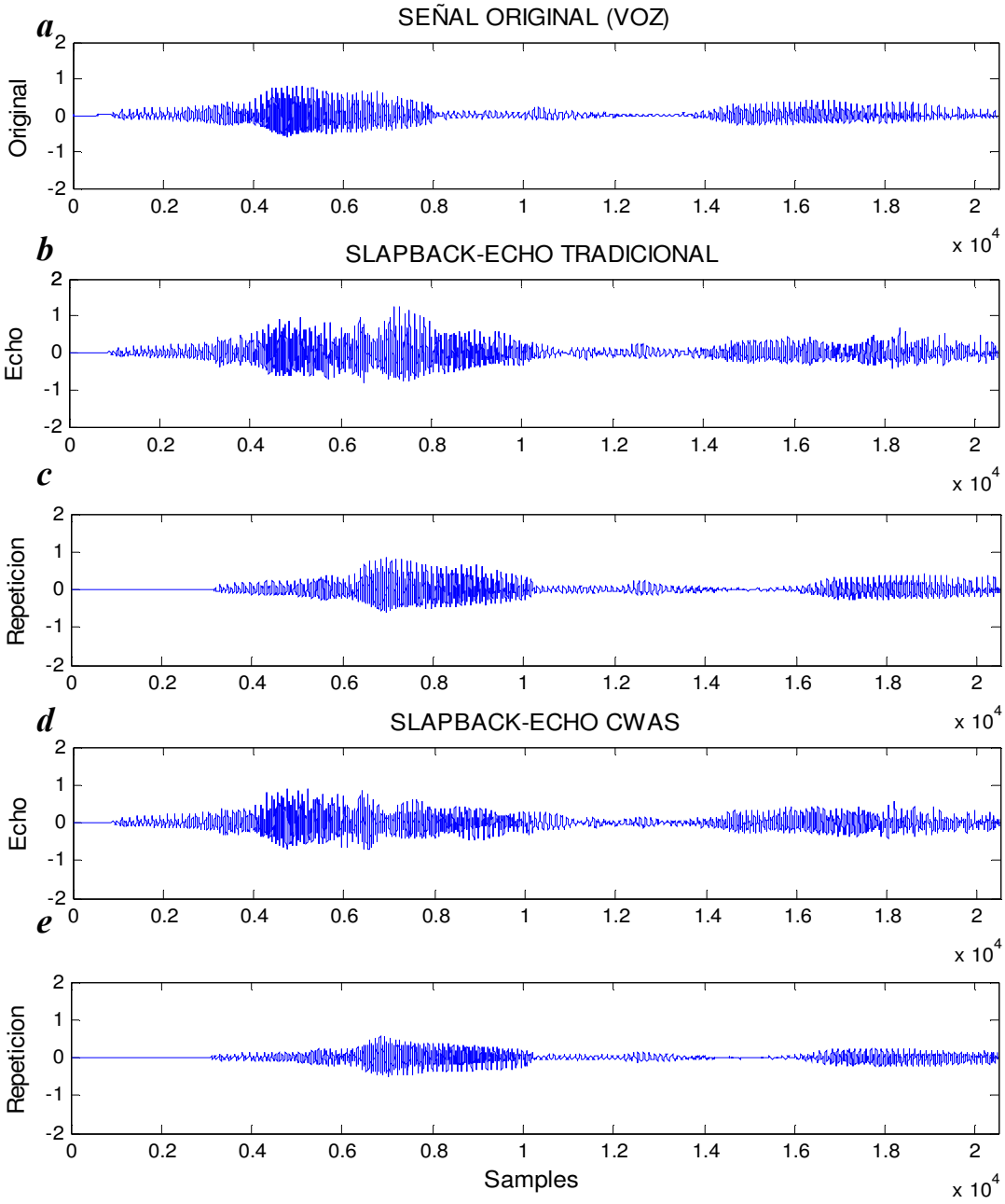


Figura 4.1. Formas de onda del efecto *Slapback-Echo* con un *delay* de 100 ms sobre una voz humana.(a) Señal original, (b) Efecto *Slapback-Echo* tradicional, (c) Señal repetida en el *Slapback-Echo* tradicional, (d) Efecto *Slapback-Echo* CWAS con parámetros g=0,7 y tn=1,01, (e) Señal repetida en el *Slapback-Echo* CWAS.

Como se puede comprobar en las Figuras 4.1(b) y 4.1(d), la aplicación del efecto *Slapback-Echo*, ya sea con el método tradicional o con el algoritmo CWAS, modifica la forma de onda del sonido original debido a la superposición que se produce entre la onda original y la onda repetida (eco). En caso de que el tiempo de *delay* elegido fuese mayor que la duración total del sonido original, esta superposición de las ondas no se produciría y el resultado del efecto sería el que se aprecia en la Figura 4.2.

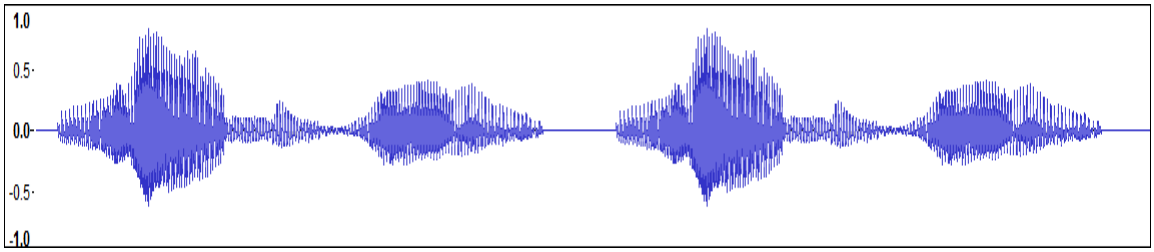


Figura 4.2. Forma de onda del efecto *Slapback-Echo* con un *delay* superior a la duración del sonido original con parámetros *g*=1 y *tn*=1.

Piano

Para apreciar mejor el efecto que la modificación del parámetro *tn* produce en el sonido final, se ha aplicado el efecto *Slapback-Echo* funcionando como *Slapback* (tiempo de *delay* de 15 ms). Para ello, se han analizado los escalogramas del sonido original (en este caso el sonido de un piano) y de los efectos *Slapback* tradicional y *Slapback* CWAS.

Comparando las Figuras 4.3(b) y 4.3(c) se comprueba que el efecto *Slapback* tradicional con un *delay* de 15 ms produce un cambio en el escalograma del sonido, realzando los armónicos más importantes pero respetando la forma del escalograma original.

Por otro lado, como se aprecia en la Figura 4.3(d), el efecto que crea el *Slapback* CWAS es la suma de dos señales que no son exactamente iguales en frecuencia, lo que en el escalograma se traduce en un ensanchamiento de cada uno de los picos del escalograma original. Este efecto puede apreciarse mejor en la Figura 4.4, en la que se ha realizado un zoom entre las frecuencias 350 y 650 Hz de los escalogramas del sonido original y del efecto *Slapback* CWAS. El escalograma del efecto *Slapback* CWAS está formado por todos y cada uno de los picos que aparecen en el escalograma del sonido original más esos mismos picos desplazados en frecuencia según determina el parámetro *tn*.

Acústicamente se puede escuchar que, mientras el *Slapback* tradicional solo realza algunos armónicos del sonido del piano y supone un cambio leve del sonido original, el *Slapback* CWAS modifica las frecuencias del sonido original y transforma el sonido del piano, dando la sensación de que es un piano antiguo, con un sonido similar al de los pianos que aparecen en las películas del oeste. Obviamente, si no se quiere conseguir este efecto y se busca un *Slapback* similar al tradicional, basta con asignar el valor 1 a los parámetros *tn* y *g*.

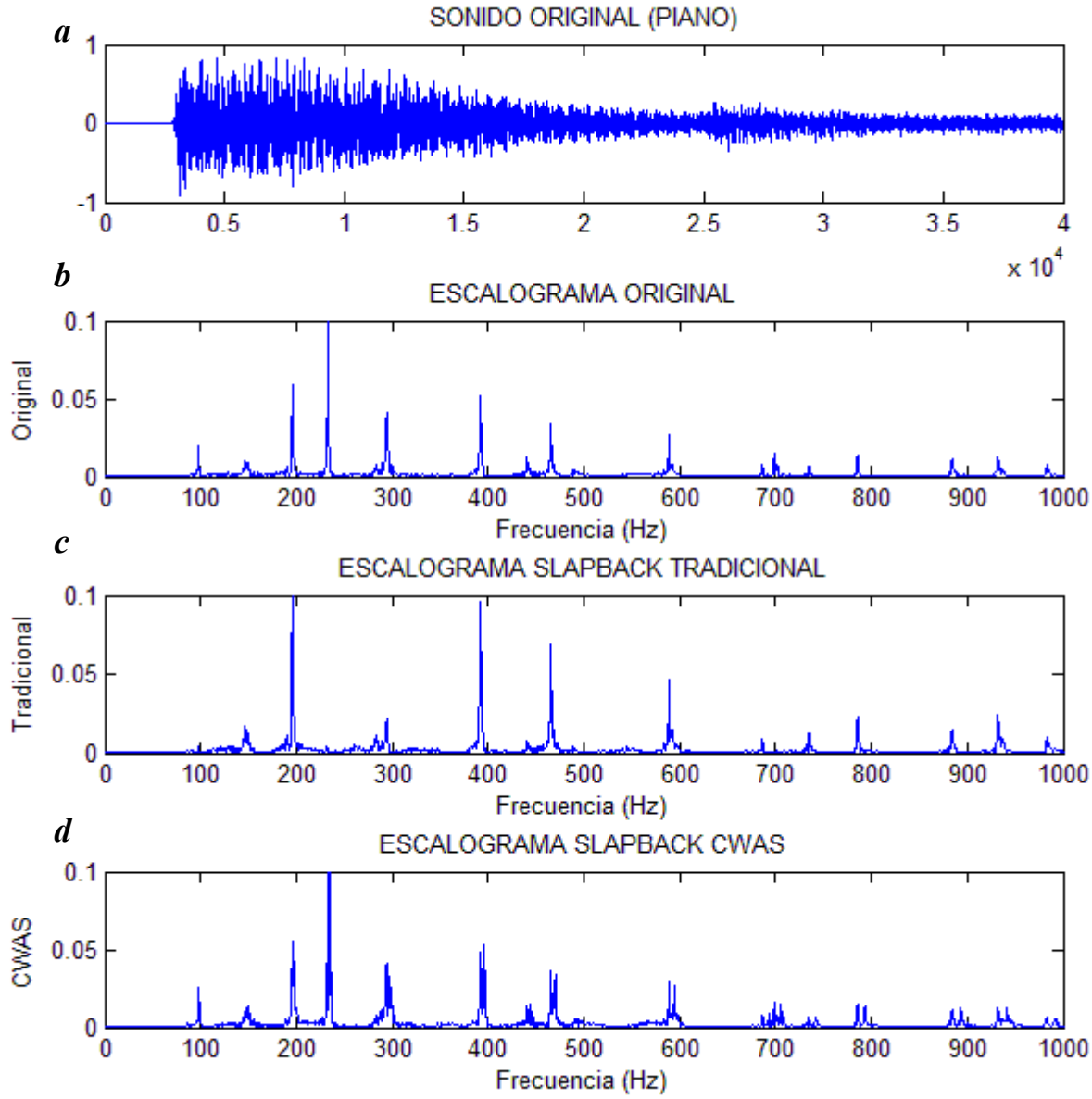


Figura 4.3. Efecto *Slapback-Echo* funcionando como *Slapback* sobre el sonido de un piano con *delay* de 15 ms. (a) Forma de onda del sonido original, (b) Escalograma del sonido original, (c) Escalograma del efecto *Slapback* tradicional, (d) Escalograma del efecto *Slapback* CWAS con g=1 y tn=1.001.

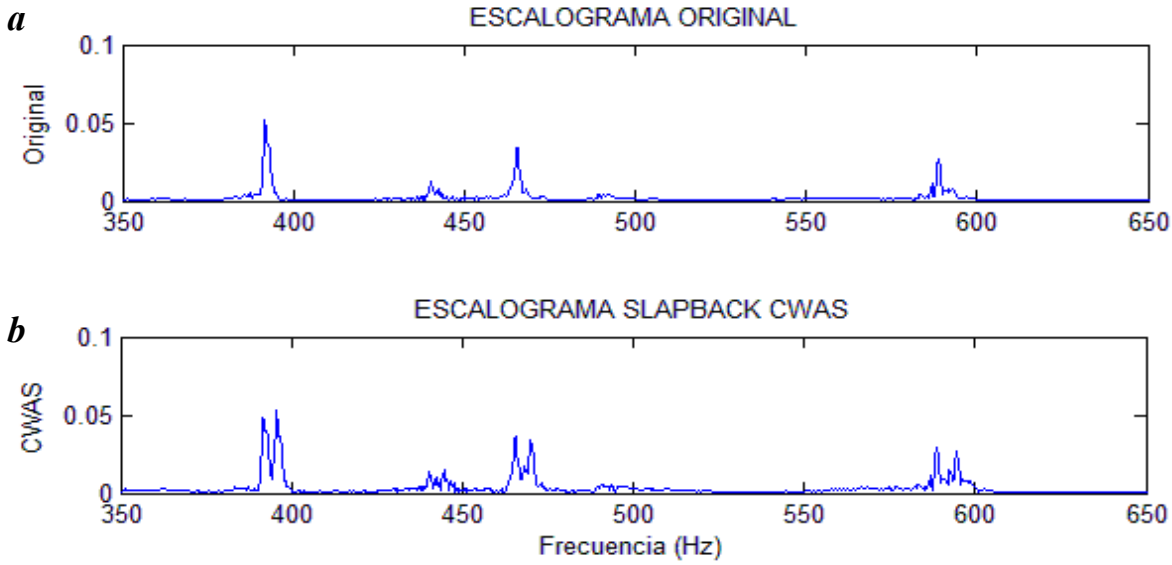


Figura 4.4. Escalogramas ampliados. (a) Señal del sonido original, (b) Efecto *Slapback* CWAS.

4.2. CHORUS

A diferencia del efecto *Chorus* tradicional, el *Chorus* CWAS permite introducir una pequeña variación en el tono en cada una de las voces que se añaden, de manera que cada una de ellas puede sonar diferente, dando la sensación de que realmente se está tocando con varios instrumentos a la vez o varias personas están cantando o hablando simultáneamente.

Con el efecto *Chorus* tradicional, no puede conseguirse esta pequeña variación tonal, por lo que cada una de las voces del sonido es una copia exacta del sonido original. Este hecho da cierto aire sintético al efecto, ya que es prácticamente imposible encontrar dos personas que hablen con el mismo tono de voz o dos instrumentos afinados exactamente igual.

En la Figura 4.5 se muestran los resultados obtenidos al aplicar un *Chorus* con dos voces (la señal original más otras dos señales) sobre un fragmento de voz humana hablada. A simple vista se puede apreciar que, aunque las formas de onda correspondientes a los efectos *Chorus* tradicional y *Chorus* CWAS respetan la silueta de la forma de onda original, son diferentes entre sí. La forma de onda del *Chorus CWAS* presenta una mayor amplitud que la del *Chorus* tradicional.

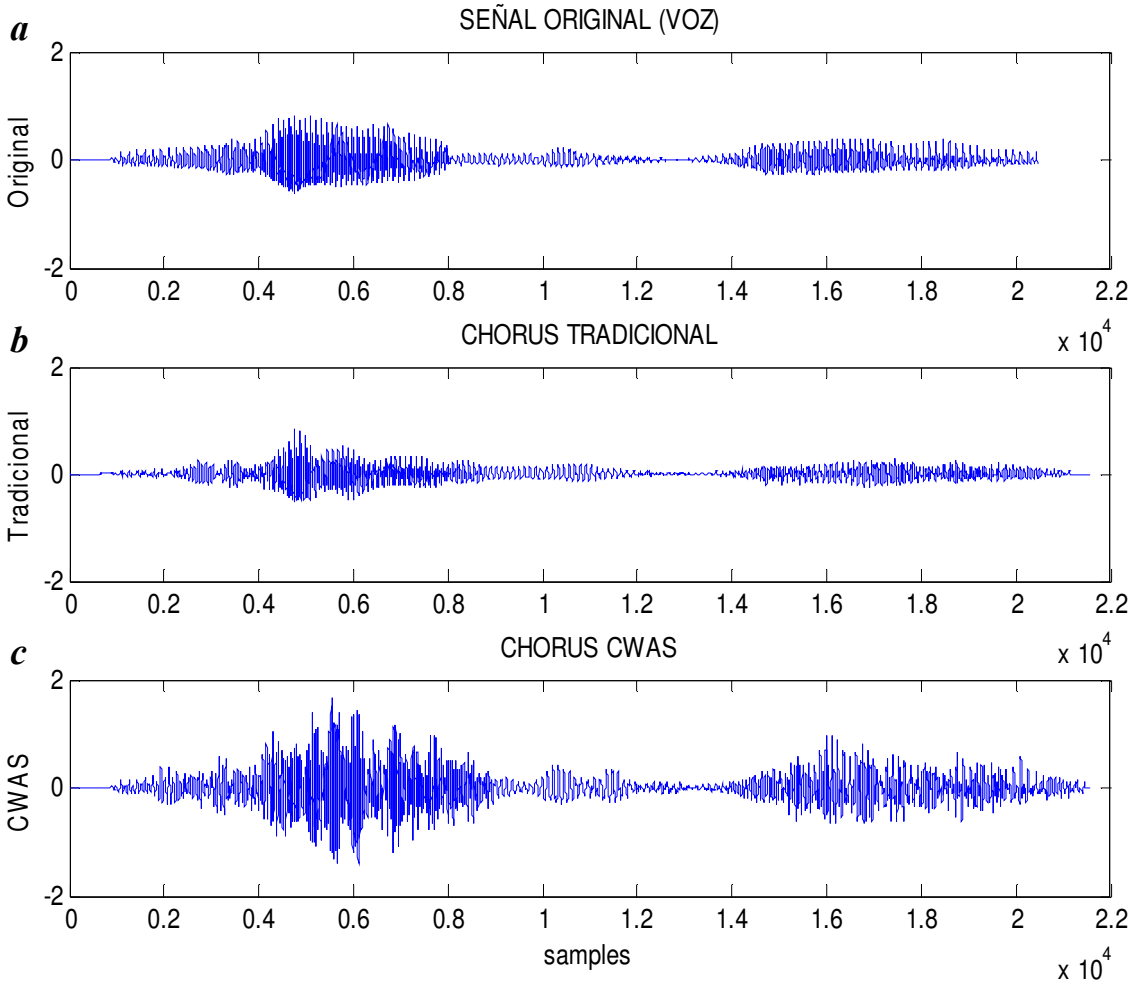


Figura 4.5. Formas de onda del efecto *Chorus*. (a) Señal original de la voz, (b) Efecto *Chorus* tradicional, (c) Efecto *Chorus* CWAS.

En la Figura 4.6 se comparan los escalogramas correspondientes a la señal original y a las señales que se obtienen al aplicar los efectos *Chorus* tradicional y *Chorus* CWAS. Los escalogramas de la señal original y del *Chorus* tradicional son prácticamente iguales ya que, a efectos de frecuencia, existe una única voz (en el *Chorus* tradicional no hay cambio frecuencial de las voces). Sin embargo, el escalograma del *Chorus* CWAS presenta cambios significativos con respecto al original. Para apreciar mejor estos cambios se ha realizado un zoom de ambos escalogramas entre las frecuencias 300 y 900 Hz (Figura 4.7). En el escalograma correspondiente al efecto *Chorus* CWAS se observan perfectamente cada una de las voces del sonido resultante.

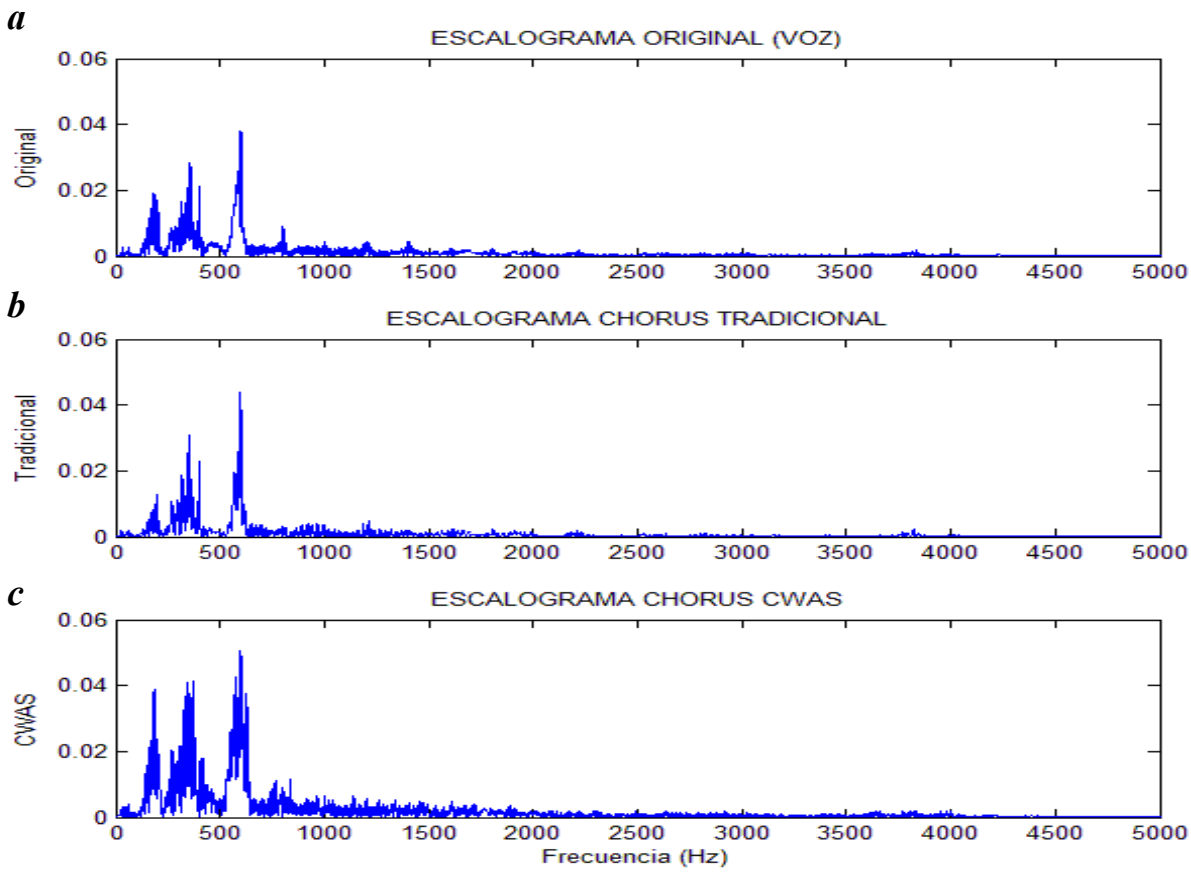


Figura 4.6. Escalogramas del efecto *Chorus*. (a) Señal original de la voz, (b) Efecto *Chorus* tradicional, (c) Efecto *Chorus* CWAS.

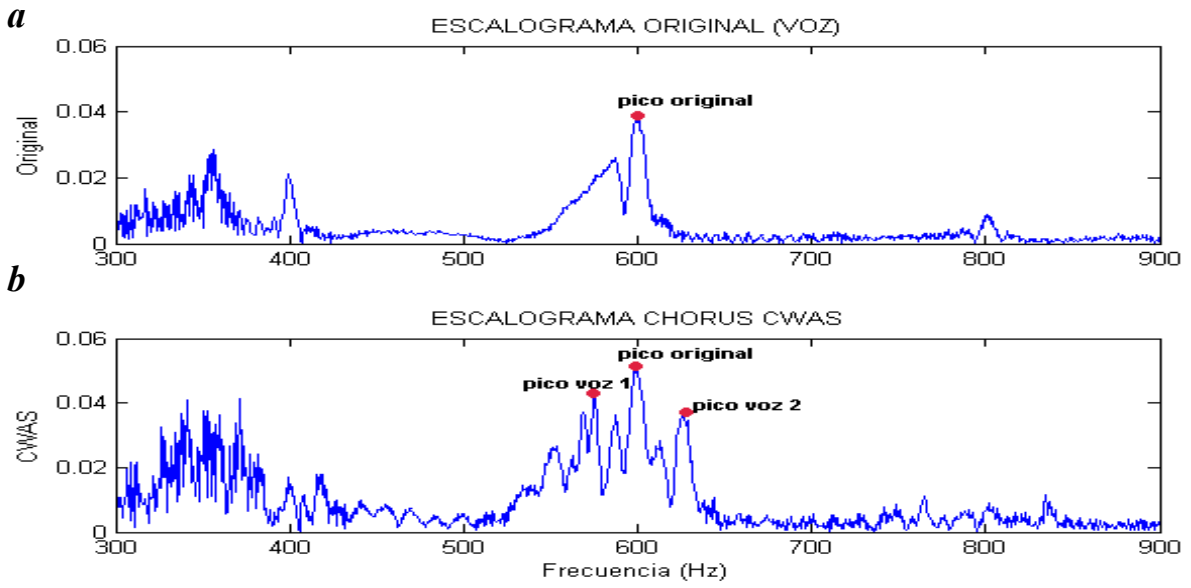


Figura 4.7. Escalogramas ampliados. (a) Señal original de la voz, (b) Efecto *Chorus* CWAS.

4.3. VIBRATO

El efecto *Vibrato* implementado a través del algoritmo CWAS ofrece idénticos resultados a los que se obtienen en un *Vibrato* tradicional, pero con un algoritmo mucho más sencillo.

En la Figura 4.8 se muestran los resultados obtenidos después de aplicar el *Vibrato* tradicional y el *Vibrato* CWAS al mismo sonido, en este caso un sonido de clarinete. Aunque las gráficas no son completamente idénticas, el resultado obtenido es muy similar en ambos casos y no se aprecian diferencias acústicas significativas.

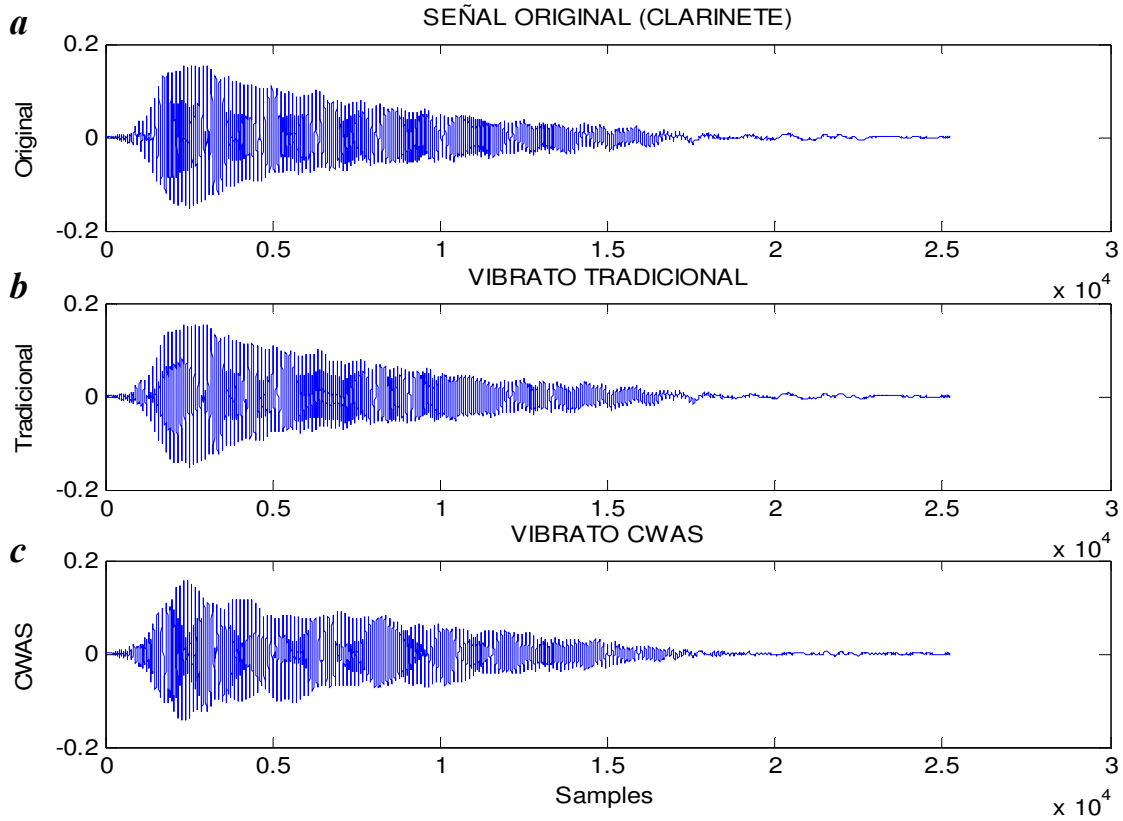


Figura 4.8. Formas de onda del efecto *Vibrato* sobre el sonido de un clarinete. (a) Sonido original, (b) Efecto *Vibrato* tradicional, (c) Efecto *Vibrato* CWAS.

La comparación de los escalogramas obtenidos a partir de los dos métodos ayuda a comprobar la similitud de ambos sonidos (Figura 4.9).

Ambos escalogramas son prácticamente iguales (se ha incluido únicamente el rango de frecuencias de 0 a 1000 Hz para comparar con más precisión ambos sonidos). Acústicamente tampoco se puede distinguir entre el *Vibrato* tradicional y el *Vibrato* CWAS ya que son prácticamente idénticos.

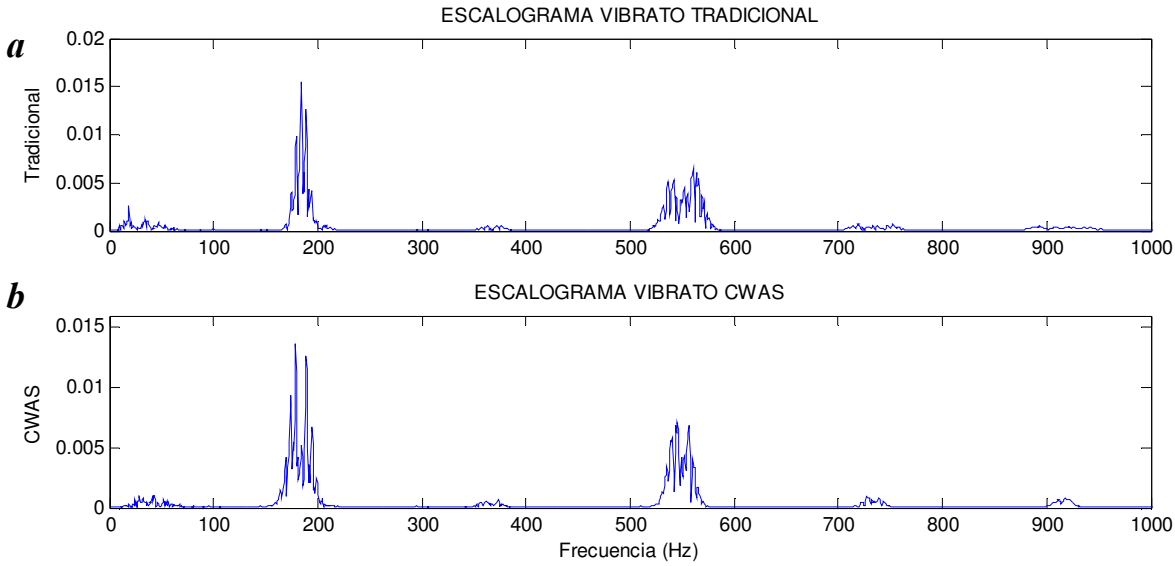


Figura 4.9. Escalogramas del efecto *Vibrato*. (a) Efecto *Vibrato* tradicional, (b) Efecto *Vibrato* CWAS.

4.4. TIME-STRETCHING

El efecto *Time-stretching* CWAS presenta evidentes mejoras respecto al *Time-stretching* tradicional. Observando las formas de onda de la Figura 4.10 se puede apreciar como el efecto tradicional no respeta la forma de onda del sonido original, mientras que con el efecto CWAS la onda original y la onda resultante del efecto tienen el mismo perfil y lo único que cambia es su extensión temporal o en *samples*. Por tanto, queda patente que la principal ventaja del *Time-stretching* CWAS es su mayor fidelidad respecto al sonido original.

Desde el punto de vista acústico también se aprecia como el efecto *Time-stretching* CWAS permite que los sonidos conserven mucho mejor sus características originales.

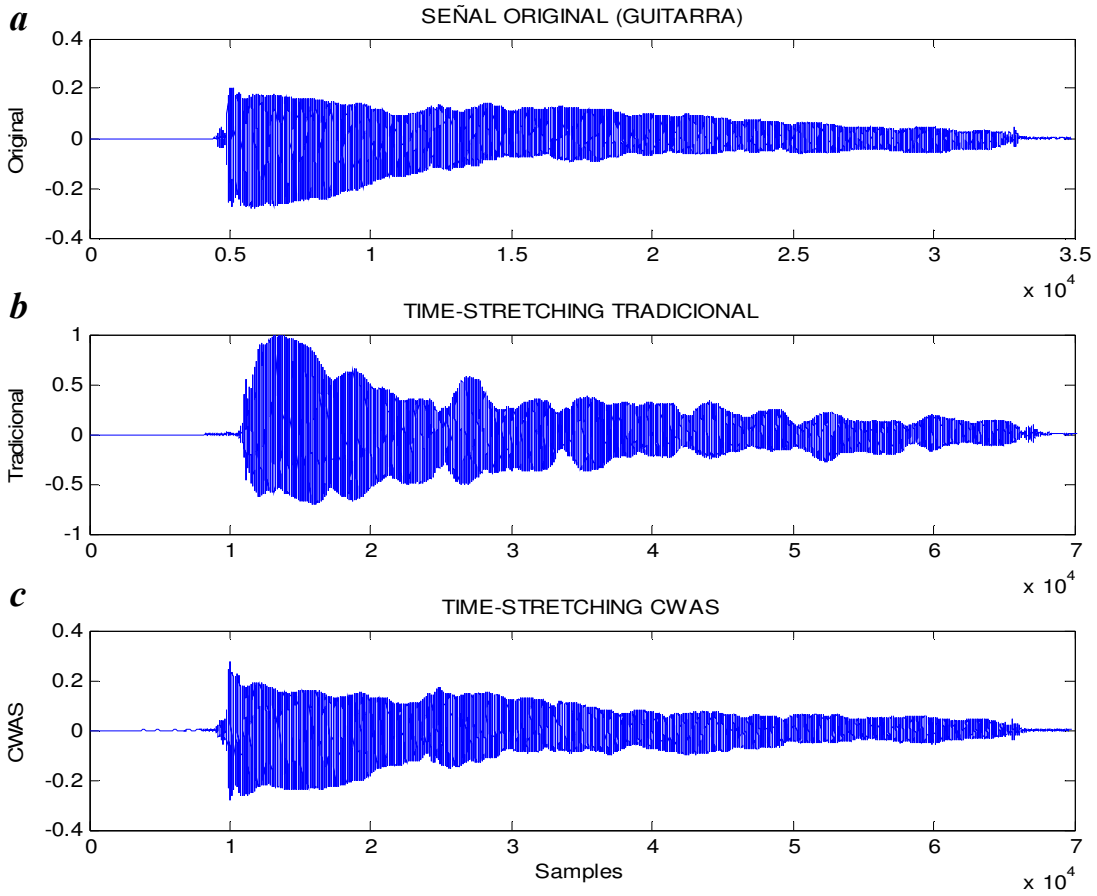


Figura 4.10. Formas de onda del efecto *Time-stretching* sobre el sonido de una guitarra en el que se ha incrementado su duración al doble de la original. (a) Sonido original, (b) Efecto *Time-stretching* tradicional, (c) Efecto *Time-stretching* CWAS.

En cuanto a la respuesta frecuencial (Figura 4.11), no se aprecian diferencias significativas y ambos métodos respetan las características frecuenciales del sonido original. En este aspecto el efecto tradicional es igual de efectivo que el efecto basado en la CWAS.

Sin embargo, a medida que se aplica un *Time-stretching* más marcado, ya sea alargando o acortando el sonido, los resultados del método tradicional empeoran, ya que la forma de onda del sonido original se deforma en exceso, disminuyendo la calidad sonora. Cuando el efecto *Time-stretching* tradicional se lleva al límite, el sonido se vuelve "borroso" y poco definido, mientras que con el método CWAS el sonido sigue conservando intactas sus características originales.

Por tanto, aunque las mejoras introducidas por el efecto CWAS son apreciables en todo el rango de funcionamiento del efecto, es en las modificaciones más marcadas cuando se puede apreciar todo su potencial.

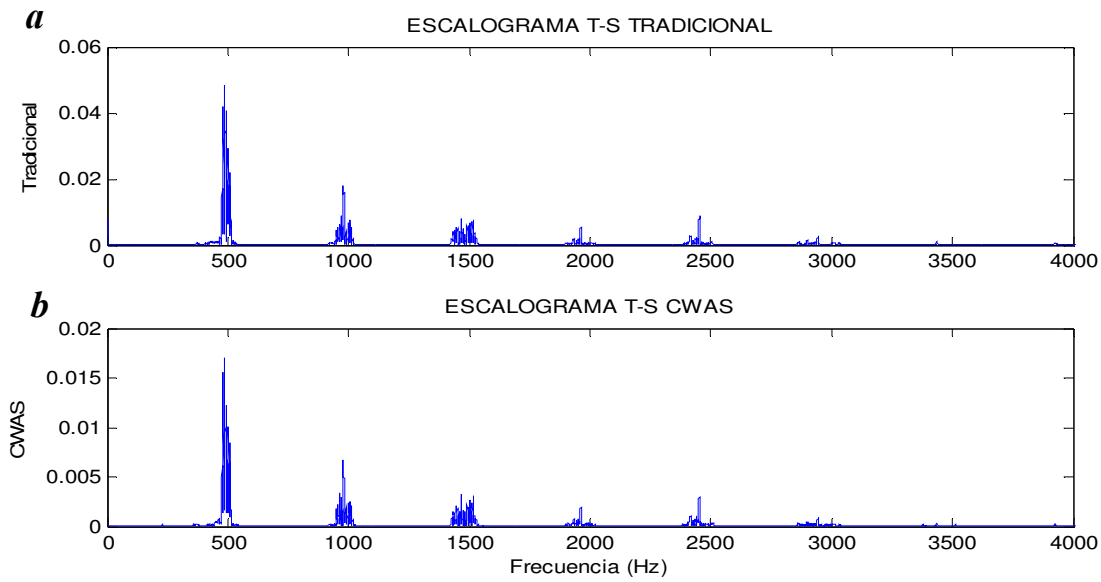


Figura 4.11. Escalogramas del efecto *Time-stretching*. (a) Efecto *Time-stretching* tradicional, (b) Efecto *Time-stretching* CWAS.

4.5. ROBOTIZACIÓN

La implementación del efecto de *Robotización* de una voz humana a partir del algoritmo CWAS ha sido una de las que más dificultades ha presentado.

Los resultados conseguidos se aproximan bastante a lo que se entiende por una *Robotización* tradicional y, acústicamente, suenan muy similares. Sin embargo, como se muestra en la Figura 4.12, las formas de onda de los sonidos obtenidos por el método tradicional y por el método CWAS no son exactamente iguales. La forma de onda obtenida mediante el método tradicional es menos respetuosa con la forma de onda del sonido original. Por el contrario, la *Robotización* CWAS presenta una forma de onda muy similar a la del sonido original.

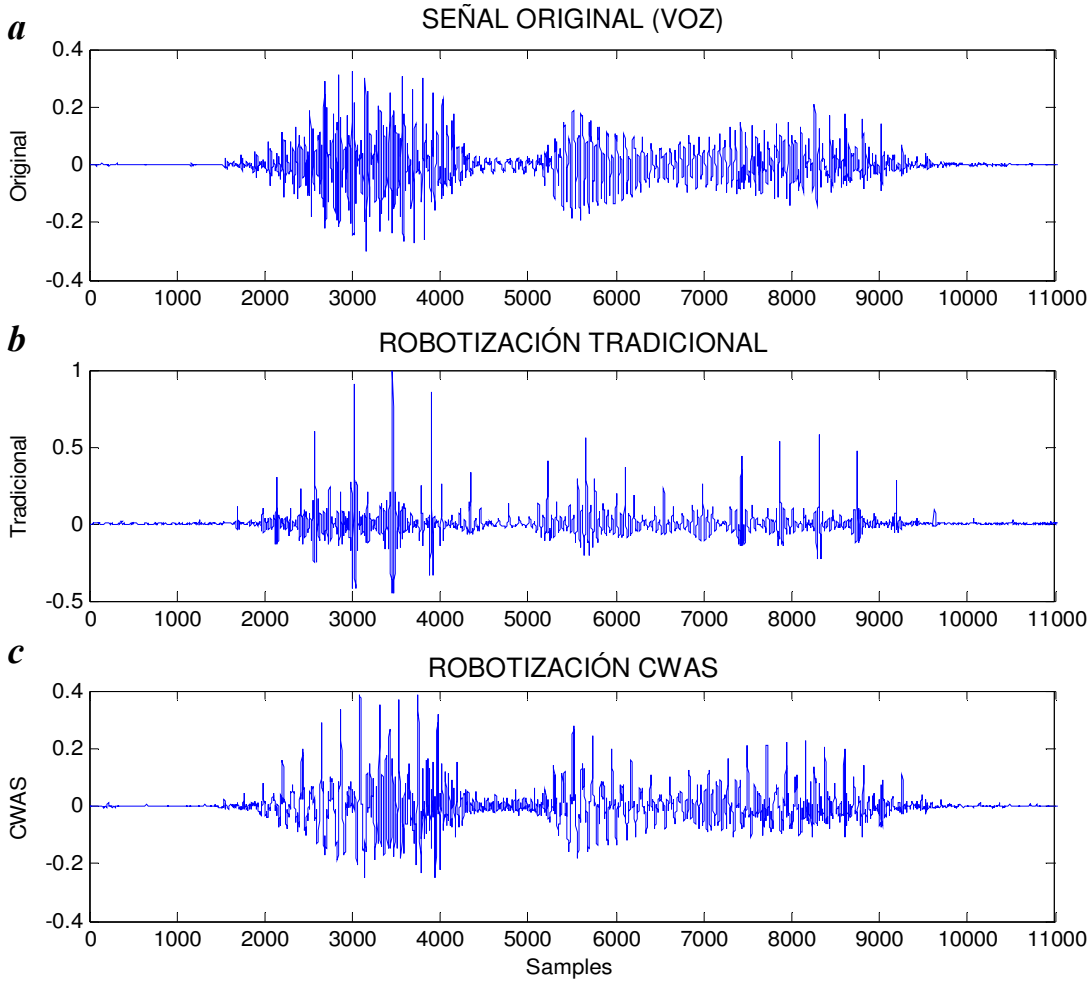


Figura 4.12. Formas de onda del efecto de *Robotización* de una voz. (a) Señal original de la voz, (b) Efecto *Robotización* tradicional, (c) Efecto *Robotización* CWAS.

En las formas de onda de los sonidos obtenidos por ambos métodos se aprecia la formación de picos que se repiten periódicamente. La distancia entre dos picos sucesivos no es la misma en los dos métodos. Esta diferencia tiene su origen en el cambio de la frecuencia de robotización ya que, a diferencia del método tradicional, el método CWAS permite elegir el tono fijo con el que se robotiza la voz.

4.6. DENOISING

La implementación del efecto *Denoising* con el algoritmo CWAS conlleva mejoras muy significativas con respecto a la aplicación del método tradicional. Es uno de los efectos en los que más se aprecia la mejora de los resultados obtenidos al comparar ambos métodos. Con el efecto *Denoising* CWAS se consigue limpiar una grabación ruidosa respetando al máximo las características del sonido original. En cambio, con el método tradicional el sonido pierde muchas de sus características originales y, por tanto, pierde calidad sonora.

El algoritmo CWAS implementado permite intervenir directamente sobre las frecuencias asociadas a ruidos y eliminar los parciales correspondientes a dichas frecuencias. Esto resulta muy útil para eliminar, por ejemplo, los ruidos producidos por la red eléctrica, que se producen a frecuencias de 50 o 60 Hz, según la región del mundo en la que se produzca la grabación.

Como se puede apreciar en la Figura 4.13, el efecto CWAS es más respetuoso con la forma de onda original que el efecto tradicional. La forma de onda "Resta CWAS" (Figura 4.13(d)) representa la parte del sonido que se ha eliminado gracias al efecto *Denoising* CWAS.

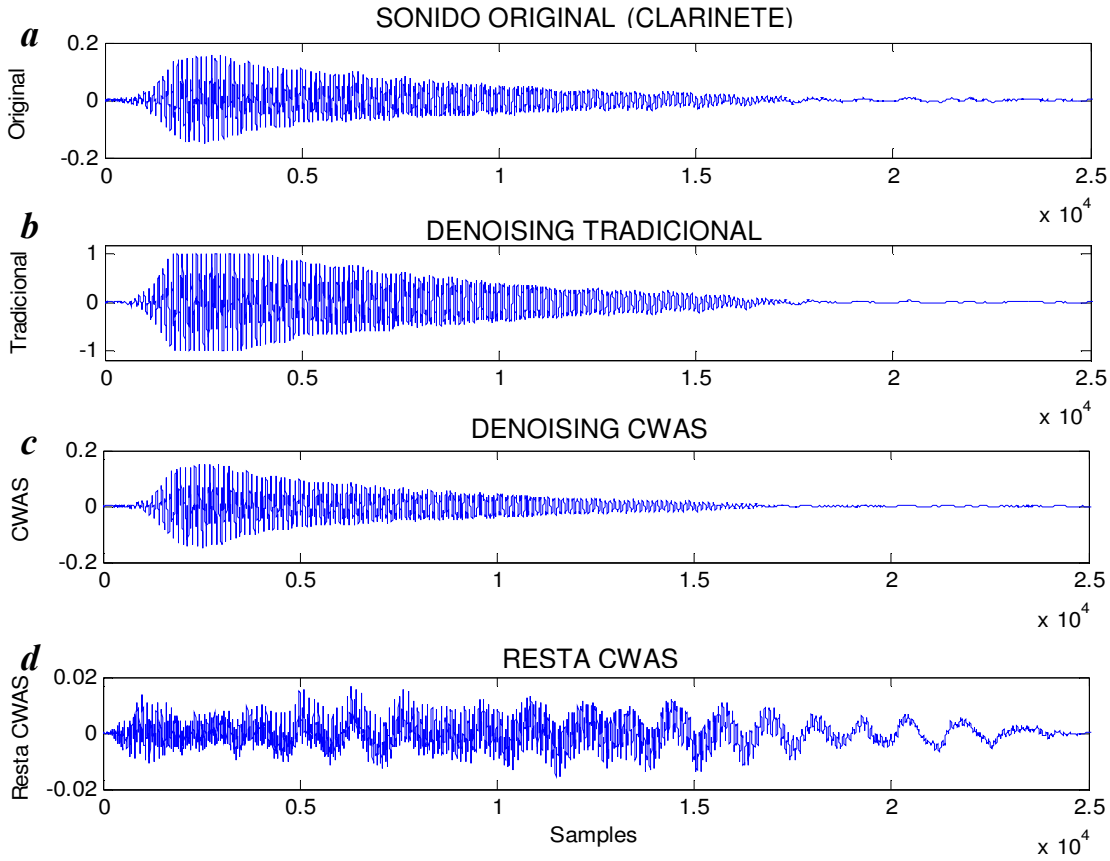


Figura 4.13. Formas de onda del efecto *Denoising* sobre el sonido de un clarinete. (a) Sonido original, (b) Efecto *Denoising* tradicional, (c) Efecto *Denoising* CWAS, (d) Resta entre el sonido original y el sonido resultante del efecto *Denoising* CWAS.

En la Figura 4.14 se muestran los escalogramas de la señal original y los resultantes de los efectos *Denoising* tradicional y *Denoising* CWAS. Se puede apreciar como el efecto tradicional modifica la respuesta frecuencial del sonido original, mientras que el efecto CWAS es mucho más respetuoso con ella.

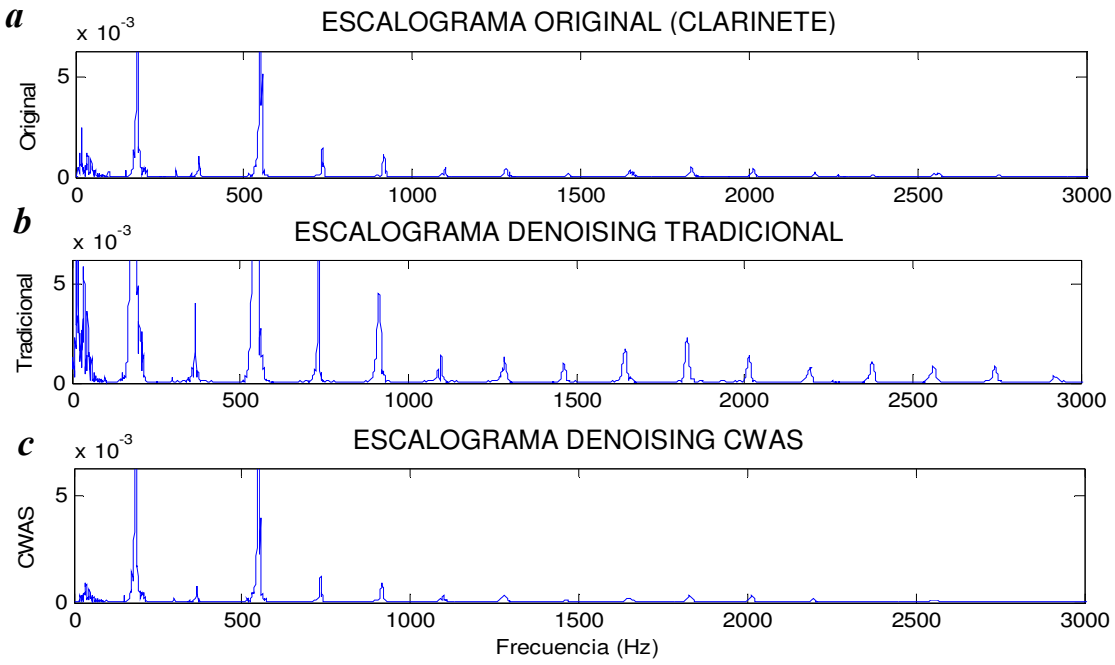


Figura 4.14. Escalogramas del efecto *Denoising*. (a) Sonido original, (b) Efecto *Denoising* tradicional, (c) Efecto *Denoising* CWAS.

4.7. PITCH-SHIFTING

Una de las características del algoritmo CWAS es que permite obtener las fases y frecuencias instantáneas de cada uno de los parciales que forman parte de un sonido, de manera que, aunque se multipliquen las frecuencias instantáneas por un factor de transposición, el sonido seguirá conservando sus cualidades originales intactas.

A diferencia del método CWAS, con el método tradicional no se obtienen datos tan exactos de las fases y frecuencias. Por tanto, si los datos de los que se parte para aplicar el *Pitch-shifting* no son precisos, el sonido resultante perderá calidad respecto al original. Esta pérdida de calidad será más evidente en un *Pitch-shifting* marcado, mientras que para un *Pitch-shifting* de unos pocos semitonos (2 o 3) el método tradicional se comporta aceptablemente bien.

A continuación se analizan los resultados obtenidos al aplicar ambos métodos para un *Pitch-shifting* de una octava (12 semitonos) por debajo del sonido original de un clarinete.

La Figura 4.15 representa las formas de onda del sonido original, del efecto *Pitch-shifting* tradicional y del efecto *Pitch-shifting* CWAS. Comparando las Figuras 4.15(a) y 4.15(b) se observa que el efecto *Pitch-shifting* tradicional con un desplazamiento de una octava no respeta la forma de onda original. La forma de onda del sonido resultante queda totalmente deformada y, acústicamente, el sonido con

efecto suena distinto del sonido original. En cambio, el efecto *Pitch-shifting* CWAS (Figura 4.15(c)) respeta totalmente la silueta de la forma de onda original.

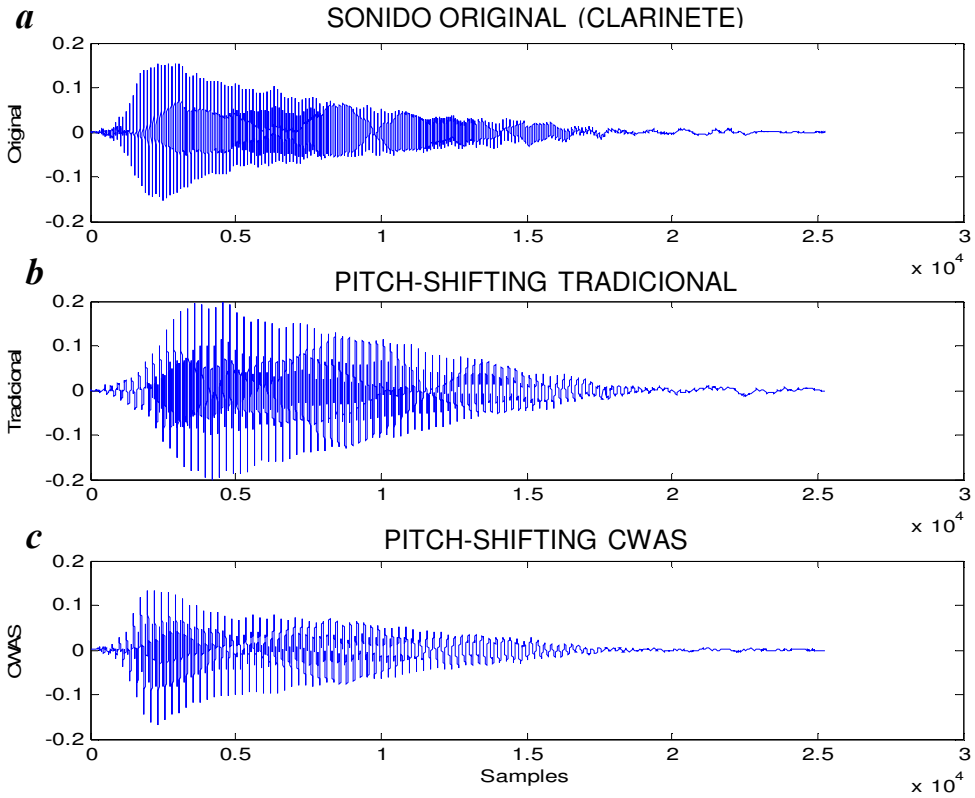


Figura 4.15. Formas de onda del efecto *Pitch-shifting* aplicando un desplazamiento de una octava por debajo del sonido de un clarinete. (a) Sonido original, (b) Efecto *Pitch-shifting* tradicional, (c) Efecto *Pitch-shifting* CWAS.

En la Figura 4.16 se muestran los escalogramas correspondientes a cada una de las formas de onda anteriores. Como se observa en dicha figura, aplicar un *Pitch-shifting* de 1 octava por debajo del sonido original equivale a reducir las frecuencias instantáneas a la mitad. Con ambos métodos (tradicional y CWAS) se obtiene el resultado esperado y cada pico del escalograma original aparece desplazado a la mitad de la frecuencia original (de 200 a 100 Hz en el caso del primer pico). Sin embargo, existe una diferencia entre ambos métodos: el escalograma del efecto CWAS respeta la proporción de los picos del escalograma original, mientras que el escalograma del efecto tradicional incluye algunos picos que no respetan dicha proporción (pico de color rojo en el escalograma).

Esto último se traduce en que, acústicamente, destacan algunos parciales que no destacaban en el sonido original, de manera que con el método tradicional se modifican las propiedades tímbricas del instrumento.

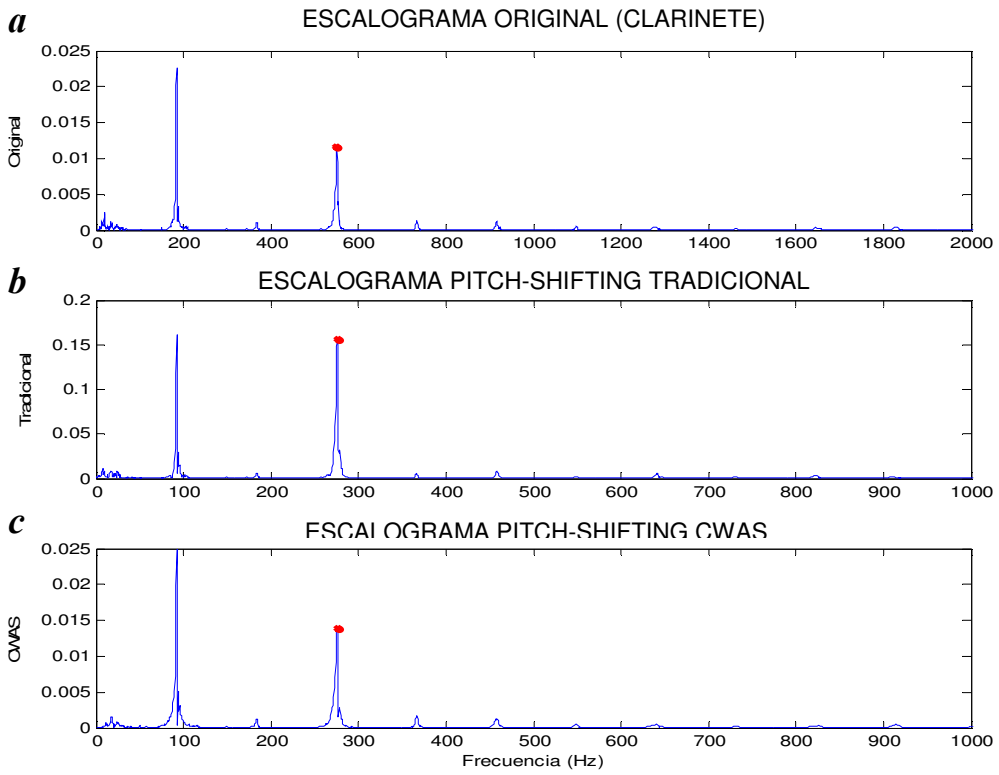


Figura 4.16. Escalogramas del efecto *Pitch-shifting*. (a) Sonido original, (b) Efecto *Pitch-shifting* tradicional, (c) Efecto *Pitch-shifting* CWAS.

En resumen, puede decirse que el efecto *Pitch-shifting* CWAS ofrece unos resultados mucho más respetuosos con la señal original que el efecto tradicional.

4.8. PITCH-SHIFTING MEJORADO

El objetivo del efecto *Pitch-Shifting mejorado* es que un sonido procedente de un instrumento o de una voz cantada pueda ser sometido a un *Pitch-shift* marcado (de al menos una octava por encima o debajo del sonido original) y mantenga, a la vez, las cualidades sonoras que tendría esa nota si se ejecutase realmente.

Para analizar los resultados se ha elegido como base una nota Do5 ejecutada por un oboe y se ha transformado en un Do4 y en un Do6 con los efectos *Pitch-shifting simple* y *Pitch-shifting mejorado*, ambos en la versión CWAS.

En la Figura 4.17 se muestran los resultados obtenidos cuando la nota base se transporta a la nota Do4.

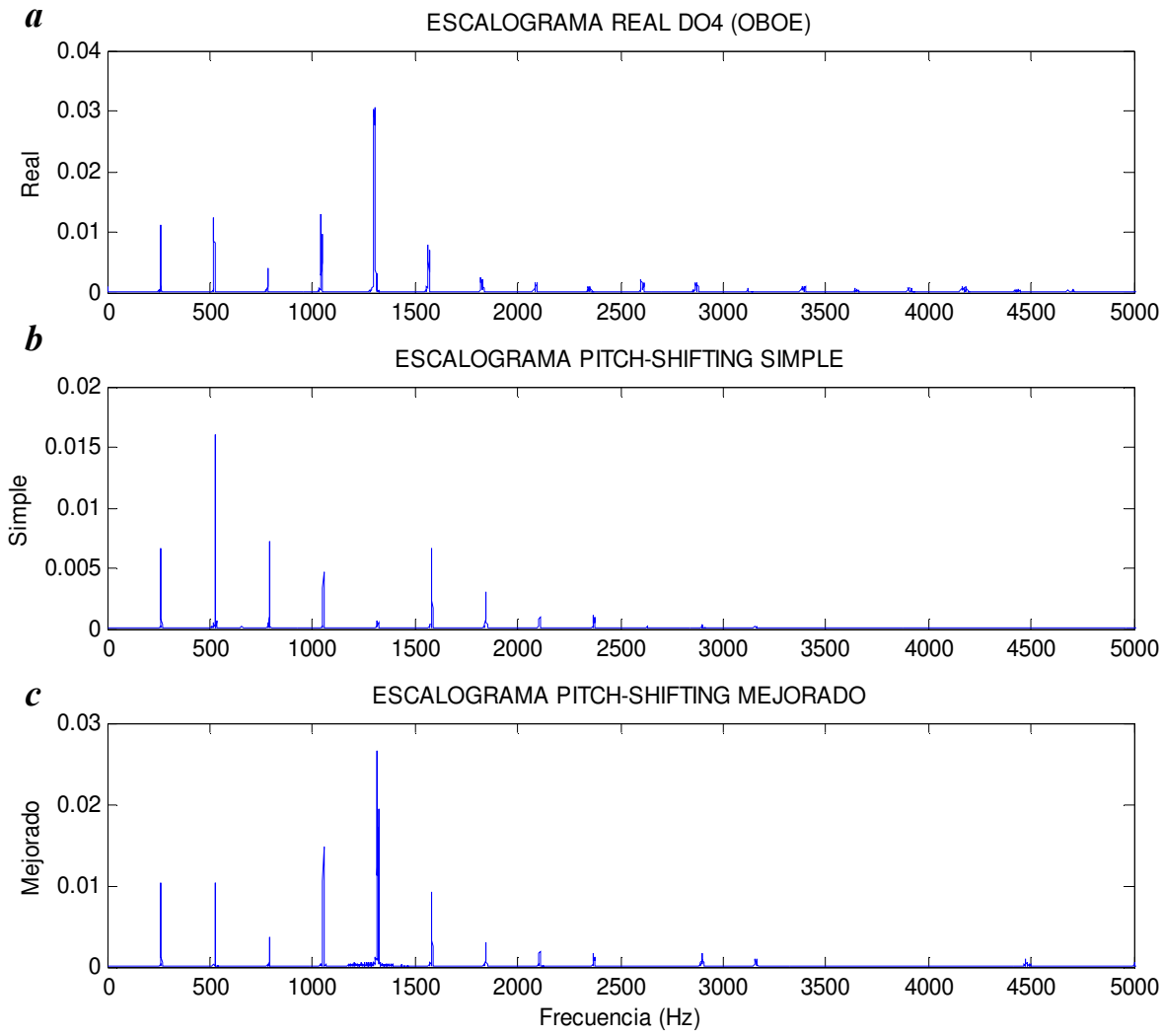


Figura 4.17. Escalogramas de la nota Do4 ejecutada por un oboe. (a) Do4 real, (b) Do4 obtenido a partir de un Do5 mediante el efecto *Pitch-shifting simple*, (c) Do4 obtenido a partir de un Do5 mediante el efecto *Pitch-shifting mejorado*.

Comparando las Figuras 4.17(a) y 4.17(b) puede observarse que el escalograma del efecto *Pitch-shifting simple* no coincide con el escalograma del sonido real, sino que son bastante diferentes. Prueba de ello es que uno de los parciales más importantes del sonido original, que se produce aproximadamente a 1250 Hz, es prácticamente inexistente en el escalograma del efecto *Pitch-shifting simple*.

Sin embargo, si se comparan los escalogramas del sonido real y del efecto *Pitch-shifting mejorado* (Figuras 4.17(a) y 4.17(c) respectivamente), se observa que hasta los 3500 Hz ambos escalogramas son prácticamente idénticos. A partir de los 3500 Hz aparecen algunas diferencias debidas a que los parciales que deberían realzarse en el sonido procedente del *Pitch-shifting mejorado* no existen en el sonido que se ha tomado como base. Aunque la nota real y la nota del efecto *Pitch-shifting mejorado* no son completamente iguales, ambas notas suenan mucho más parecidas que la que

se obtiene del efecto *Pitch-shifting simple* y las diferencias acústicas entre ambas son casi inapreciables.

En la Figura 4.18 se muestran los resultados obtenidos cuando la nota base se transporta a la nota Do6. De nuevo se observa como el escalograma correspondiente al *Pitch-shifting simple* es bastante diferente del original, mientras que el escalograma del *Pitch-shifting mejorado* es prácticamente idéntico al original.

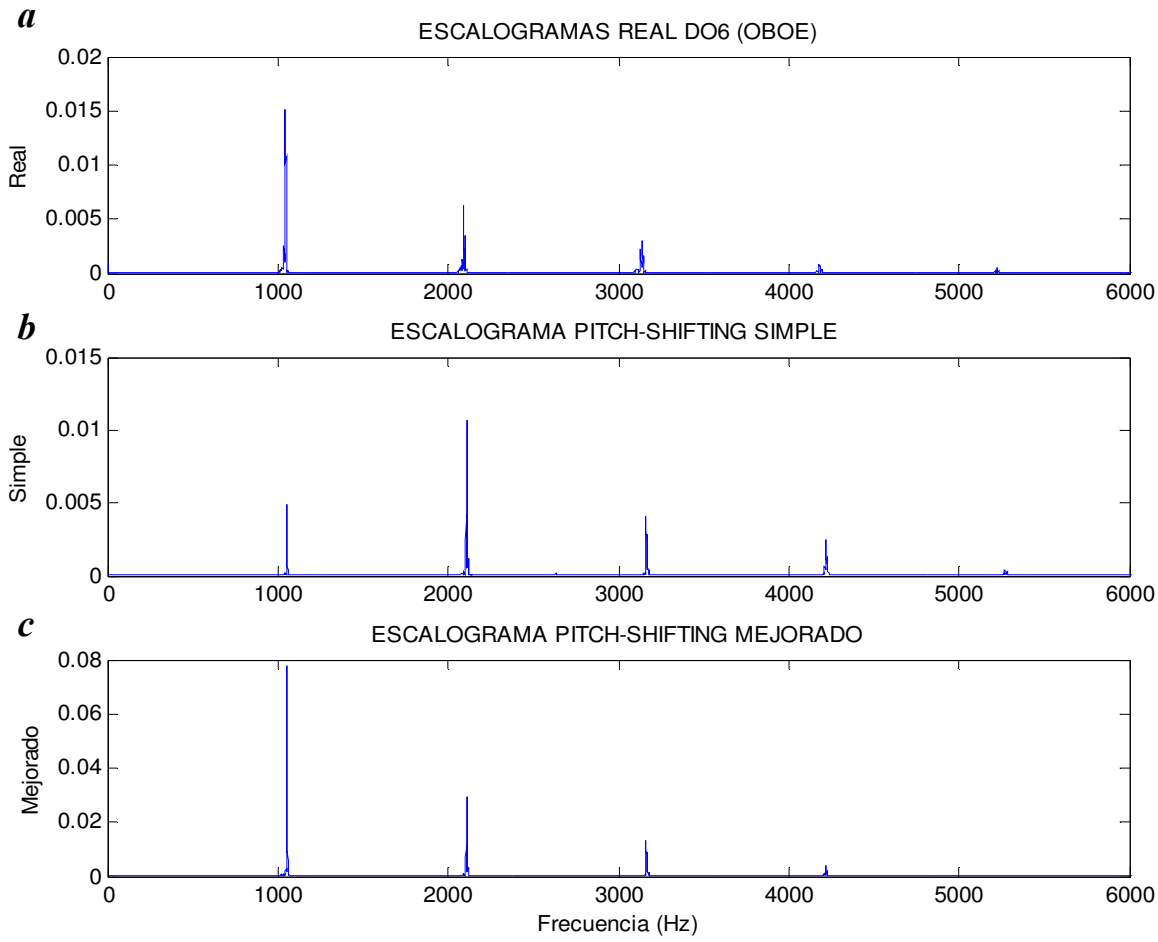


Figura 4.18. Escalogramas de la nota Do6 ejecutada por un oboe. (a) Do6 real, (b) Do6 obtenido a partir de un Do5 mediante el efecto *Pitch-shifting simple*, (c) Do6 obtenido a partir de un Do5 mediante el efecto *Pitch-shifting mejorado*.

Los resultados que ofrece el efecto *Pitch-shifting mejorado* son muy satisfactorios ya que acústicamente se consiguen sonidos muy similares a las notas reales.

4.9. SUSTAIN

El objetivo del efecto *Sustain* es encontrar la región estable de un sonido (región sustain) y alargar la duración de dicho tramo según las preferencias del usuario. Este efecto ha sido implementado por primera vez en este proyecto, por lo que no ha sido posible realizar una comparación con la implementación tradicional.

Para que el efecto *Sustain* funcione correctamente, es necesario que el perfil de volumen del sonido original se corresponda con la envolvente ADSR. En caso contrario, es posible que el algoritmo CWAS planteado no localice de forma correcta la región *sustain* del sonido.

En la Figura 4.19(a) se muestra la forma de onda del sonido original (nota ejecutada por una guitarra). Como se puede observar, debido a las particularidades del sonido que produce una guitarra, resulta complicado identificar la envolvente ADSR del sonido teniendo en cuenta únicamente la forma de onda del sonido. Sin embargo, observando la Figura 4.19(b), que representa la suma de los módulos de todos los parciales que forman la nota, se localizan fácilmente las fases *attack, decay, sustain* (señalada en color rojo) y *release* del sonido.

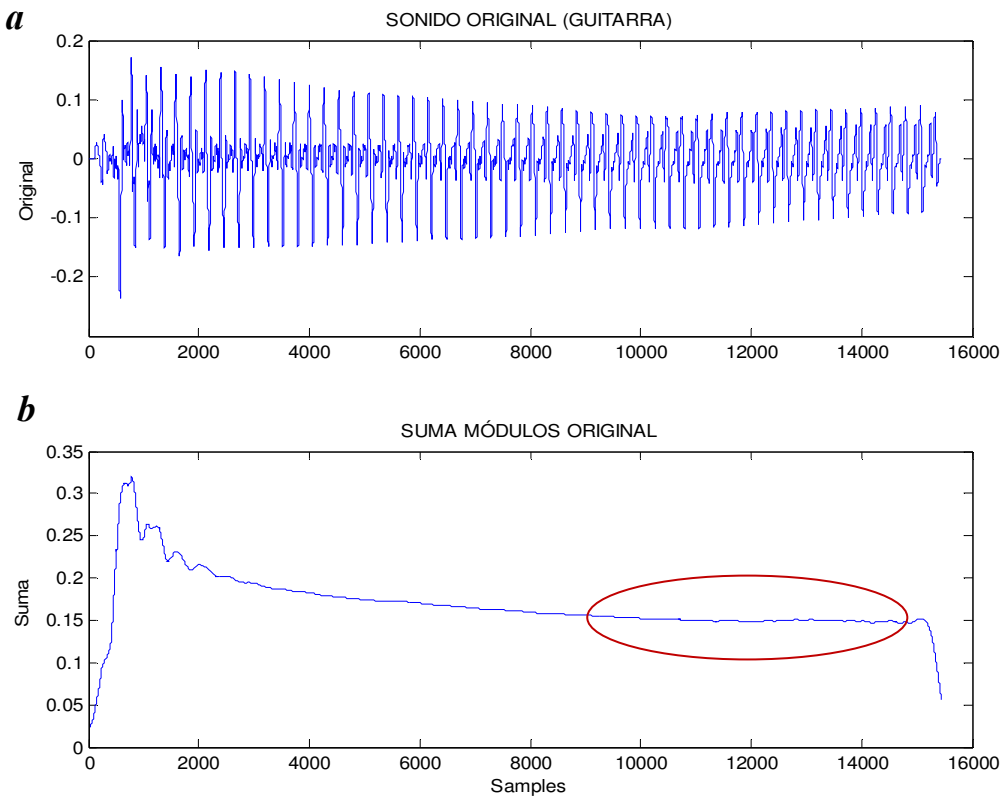


Figura 4.19. Sonido original de una nota ejecutada por una guitarra. (a) Forma de onda del sonido original, (b) Suma de los módulos de todos los parciales que forman el sonido original.

El algoritmo CWAS desarrollado en este proyecto trabaja parcial por parcial. En cada uno de ellos localiza un tramo de 1000 *samples* dentro de la región *sustain* que sea lo más estable posible (que sus valores varíen lo mínimo posible).

En la Figura 4.20 se puede observar como el algoritmo cumple perfectamente con su función, localizando y alargando la región *sustain* del sonido. Cada tramo estable de 1000 *samples* se ha repetido un total de 30 veces y el resultado es un aumento de 30000 *samples* en la duración total del sonido original.

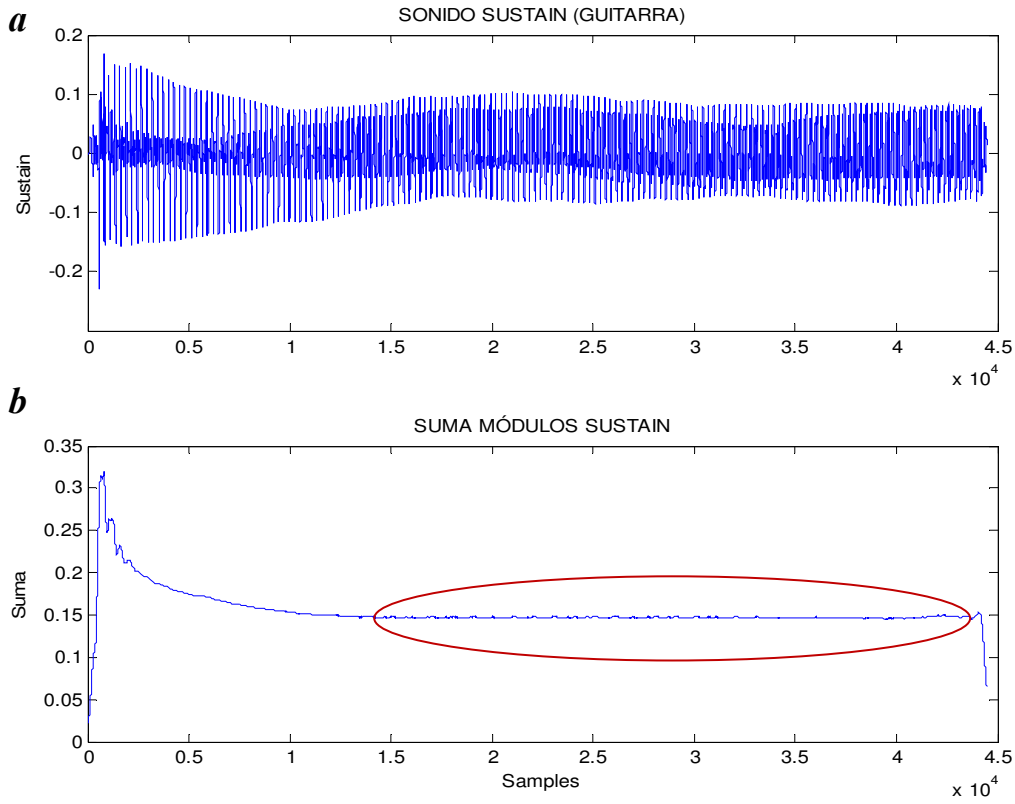


Figura 4.20. Sonido resultante de aplicar el efecto *Sustain* a una nota ejecutada por una guitarra. (a) Forma de onda del sonido, (b) Suma de los módulos de todos los parciales que forman el sonido.

Comparando las Figuras 4.19(b) y 4.20(b) puede observarse como las regiones *attack*, *decay* y *release* no se ven alteradas por la aplicación del efecto y mantienen tanto su extensión temporal como su perfil. Sin embargo, al producirse un gran aumento de la duración de la región *sustain*, dichas regiones pierden importancia en relación al perfil del sonido.

Los resultados acústicos conseguidos con el algoritmo del efecto *Sustain* CWAS son muy satisfactorios y corroboran todos los aspectos señalados anteriormente en el análisis gráfico.

4.10. VOCODER SIMPLE

La finalidad del efecto *Vocoder simple* es aplicar la envolvente de volumen de un sonido de control sobre un sonido base.

El efecto *Vocoder simple* ha sido implementado en este proyecto a partir del algoritmo CWAS. Puesto que se desconoce el algoritmo para su implementación por los métodos tradicionales, no ha sido posible realizar una comparación directa entre ambos métodos (realmente existe una definición teórica sobre la implementación tradicional del efecto, pero en el libro DAFX consultado no aparece el *script* de *Matlab* correspondiente).

Como se puede observar en la Figura 4.21, la forma de onda del sonido resultante del efecto *Vocoder simple* es una combinación de las formas de onda de los sonidos base y de control, como si sobre la forma de onda del sonido base se recortara la forma de onda del sonido de control. Al aplicar el efecto directamente sobre los módulos de los parciales, se consigue respetar al máximo el contenido frecuencial del sonido base.

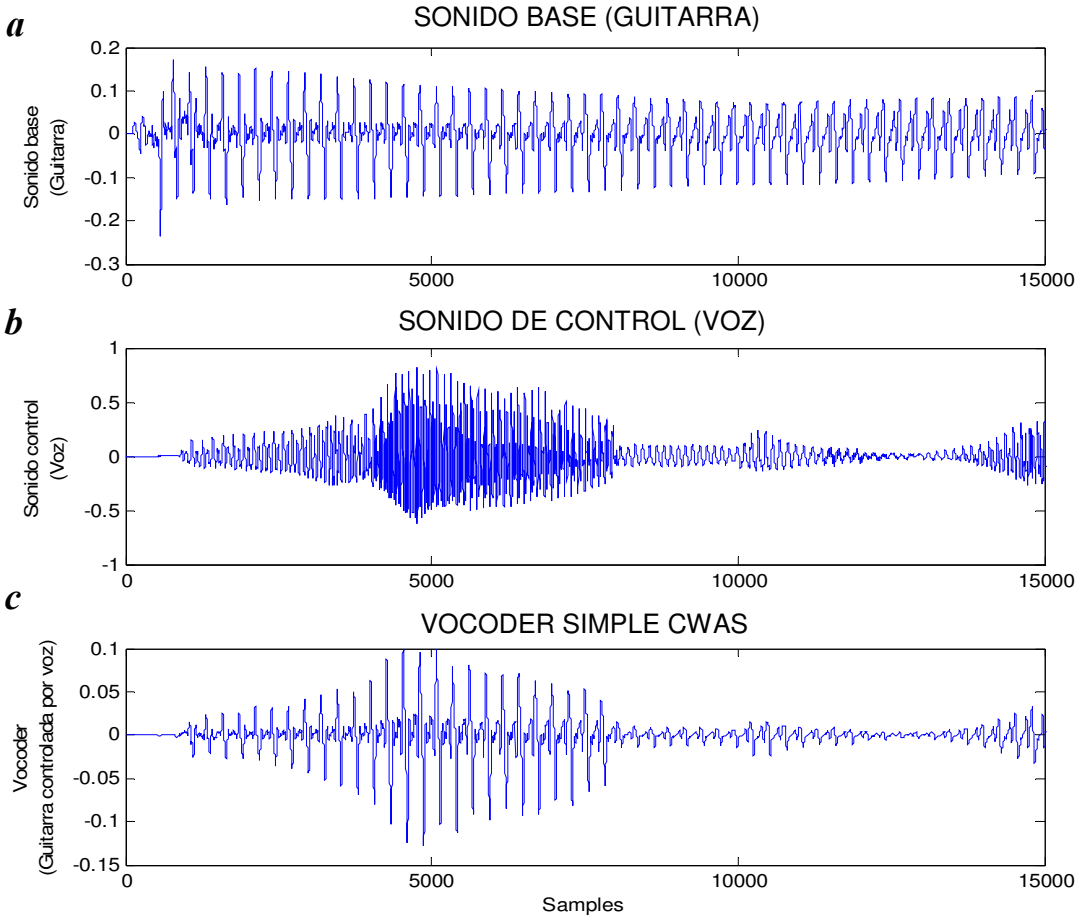


Figura 4.21. Formas de onda del efecto *Vocoder simple*. (a) Sonido base ejecutado por una guitarra, (b) Sonido de control consistente en una voz humana, (c) Efecto *Vocoder simple* CWAS.

4.11. VOCODER MEJORADO

Como ya se comentó en el capítulo anterior de la memoria, el efecto que se consigue con el *Vocoder mejorado*, siempre que el sonido base sea una única nota, es muy similar al efecto *Robotización*. El resultado es una voz que adquiere las cualidades tímbricas y el tono de una nota ejecutada por un instrumento.

A continuación se analizan los resultados obtenidos con el efecto *Vocoder mejorado* implementado a través del algoritmo CWAS.

Como se puede observar en la Figura 4.22, las formas de onda del sonido de control (voz) y del sonido resultante son prácticamente iguales. Por otro lado,

comparando los escalogramas (Figura 4.23) se comprueba que la respuesta frecuencial del sonido resultante es muy similar a la del sonido base. Los armónicos del sonido son los mismos, aunque al modificar los módulos cambia ligeramente la energía de algunos de ellos.

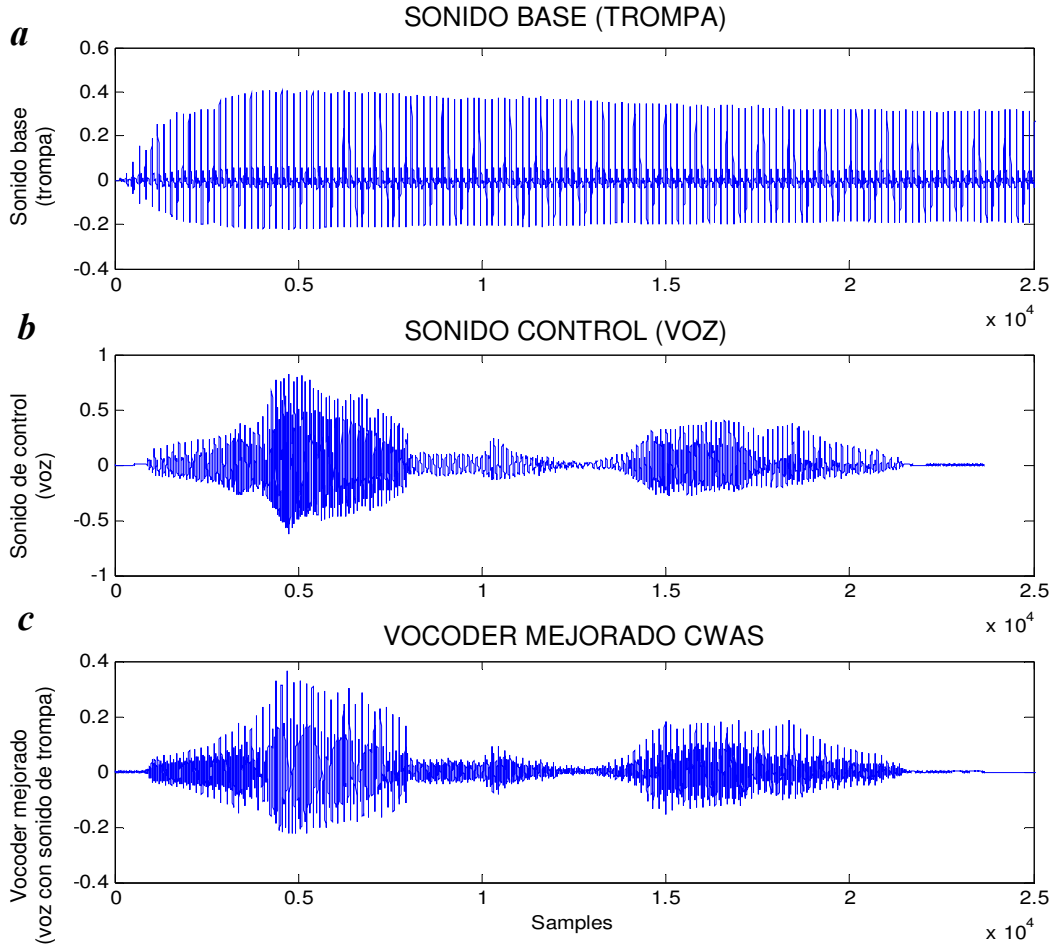


Figura 4.22. Formas de onda del efecto *Vocoder mejorado*. (a) Sonido base ejecutado por una trompa, (b) Sonido de control consistente en una voz humana, (c) Efecto *Vocoder mejorado* CWAS.

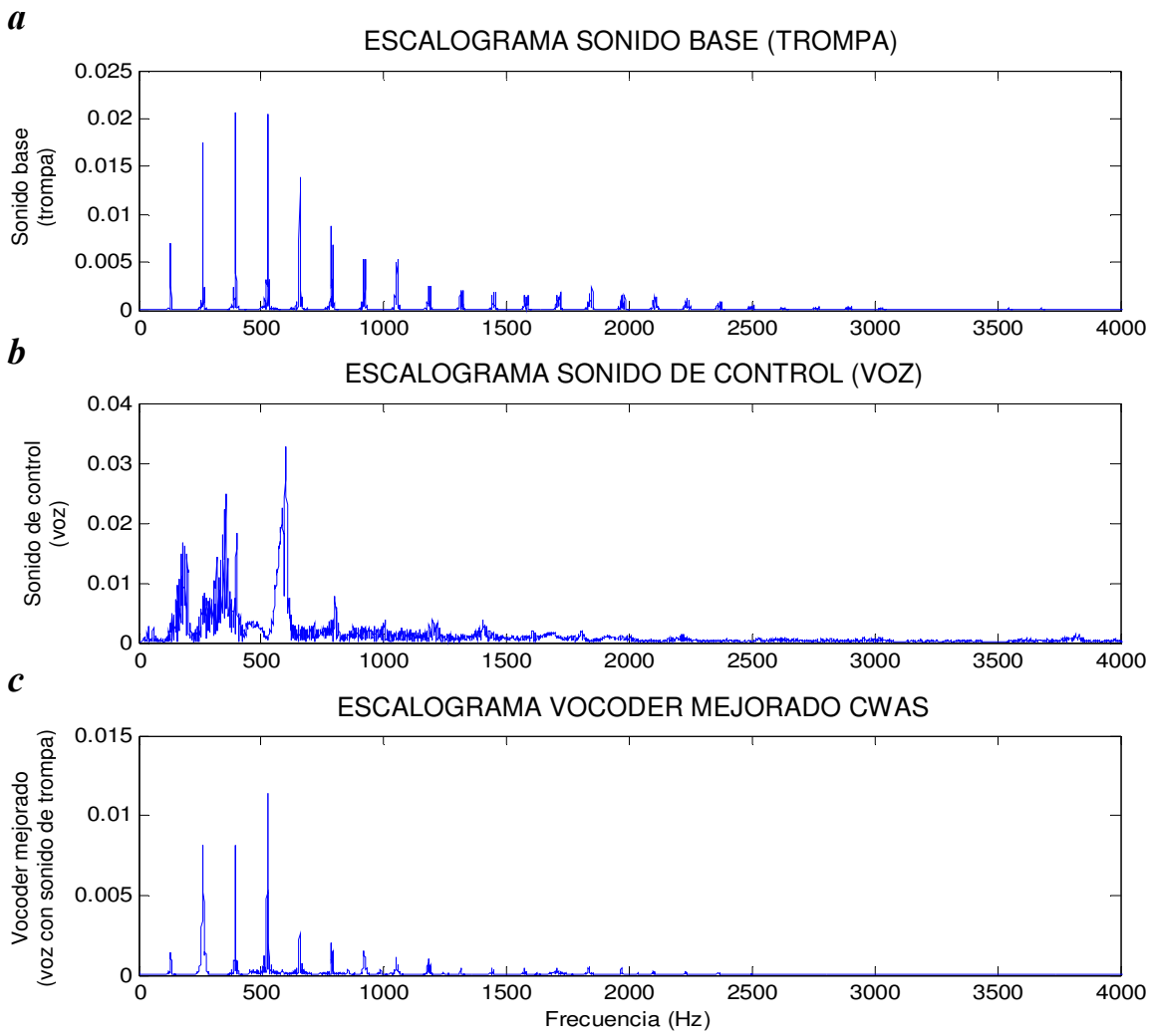


Figura 4.23. Escalogramas del efecto *Vocoder mejorado*. (a) Sonido base ejecutado por una trompa, (b) Sonido de control consistente en una voz humana, (c) Efecto *Vocoder mejorado* CWAS

Una de las principales diferencias entre los efectos *Vocoder simple* y *Vocoder mejorado* es el resultado acústico, ya que tras la aplicación del *Vocoder mejorado* se entienden perfectamente las palabras que pronuncia la voz, mientras que con el *Vocoder simple* esto no ocurre.

4.12. MORPHING

Como ya se ha comentado en el apartado 3.10, el efecto *Morphing* es un efecto del que no se conoce su implementación por los métodos tradicionales.

Para analizar el efecto *Morphing* se han utilizado varias mezclas de dos sonidos distintos procedentes de una guitarra y de un clarinete, combinando dichos sonidos en distintos porcentajes de mezcla.

Puesto que se han utilizado dos notas distintas (con distinta frecuencia fundamental) procedentes de dos instrumentos diferentes, el sonido cambia de textura

y de tono a medida que se modifican los porcentajes usados para combinar ambos sonidos.

En la Figura 4.24 se muestran las formas de onda de los sonidos resultantes tras aplicar el efecto *Morphing* CWAS con distintos porcentajes de mezcla.

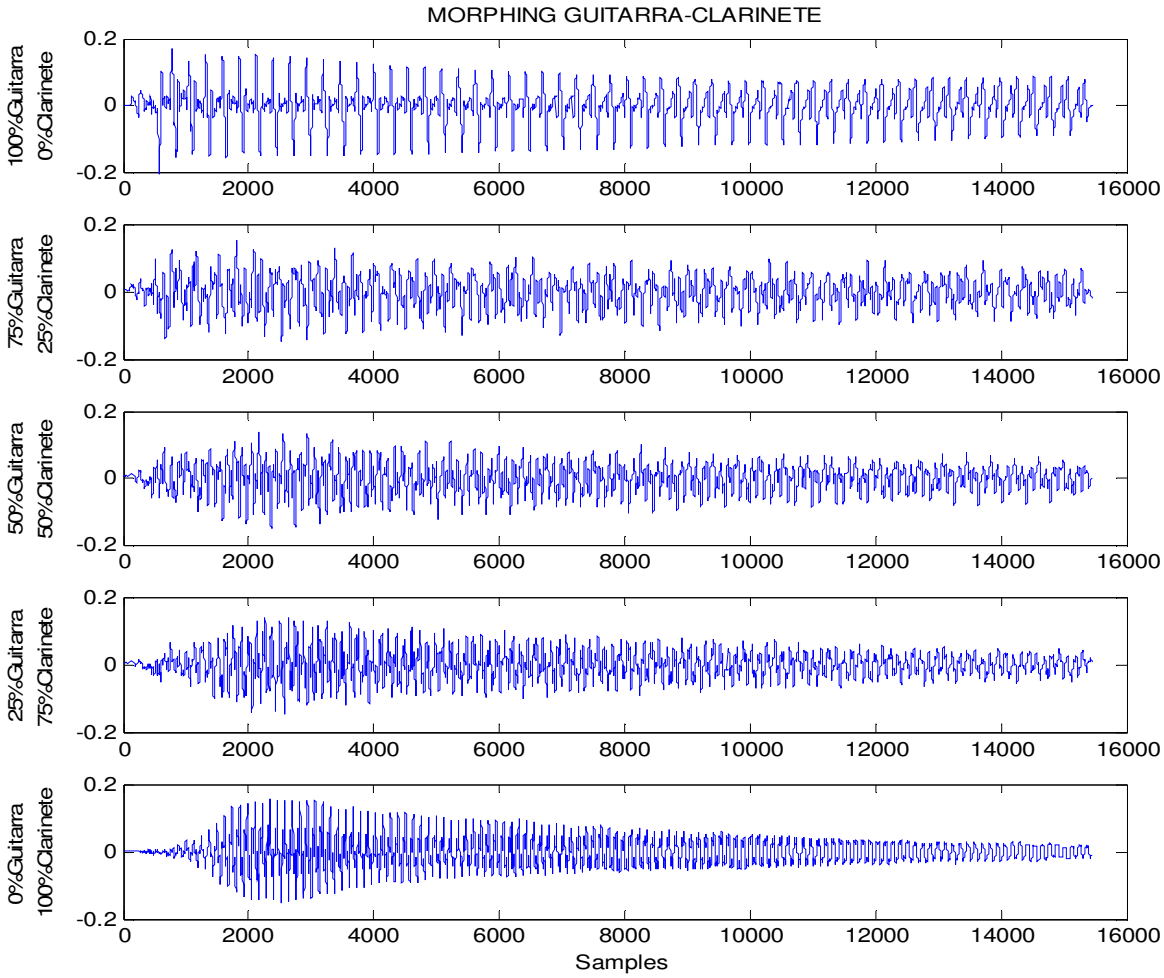


Figura 4.24. Formas de onda del efecto *Morphing* CWAS para distintos porcentajes de mezcla de dos notas ejecutadas por una guitarra y por un clarinete.

Como se puede observar en la Figura 4.24, las formas de onda de los sonidos ejecutados por la guitarra y el clarinete se van combinando según el porcentaje de mezcla de ambos. Por ejemplo para una combinación del 25% de guitarra y 75% de clarinete, la forma de onda del sonido resultante sigue el contorno de la forma de onda del sonido del clarinete, pero conservando a la vez algunas propiedades del sonido de la guitarra.

Si se analizan los escalogramas obtenidos para cada porcentaje de mezcla (Figura 4.25) se puede observar como los distintos picos de los escalogramas se van desplazando para adaptarse al sonido final. Si, por ejemplo, se analiza el primer pico señalado en rojo, se observa como a medida que disminuye el porcentaje de guitarra

se va desplazando desde los 80 Hz hasta los 33 Hz y, a su vez, va disminuyendo su energía de forma progresiva hasta que en el escalograma correspondiente al 100% de clarinete prácticamente desaparece.

Otro aspecto que puede destacarse es que, puesto que se están combinando dos sonidos con distintas frecuencias fundamentales, la distancia entre picos sucesivos va cambiando, ampliándose a medida que aumenta la presencia del sonido del clarinete.

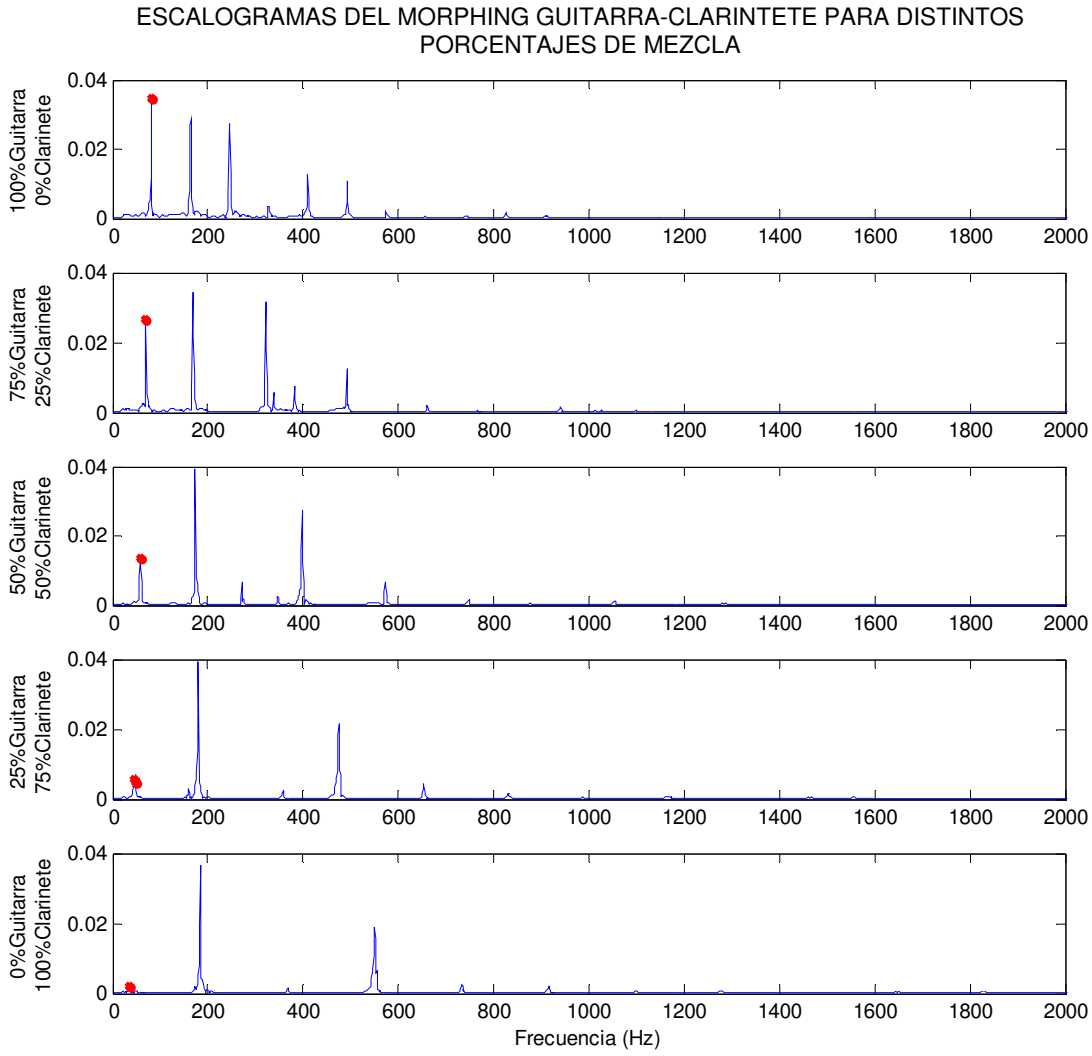


Figura 4.25. Escalogramas del efecto *Morphing* para distintos porcentajes de mezcla.

4.13. PSEUDOROBOT

El efecto *Pseudorobot* ha surgido en un intento de implementar el efecto de *Robotización* tradicional mediante la técnica CWAS. Los resultados obtenidos son similares a los de la *Robotización* tradicional, pero existen varias diferencias que hacen que el efecto no se pueda considerar exactamente como una *Robotización* tradicional.

A continuación se comparan los resultados obtenidos con el efecto *Pseudorobot* y con la *Robotización* tradicional para apreciar mejor las diferencias y similitudes entre ambos efectos.

Como se puede observar en la Figura 4.26, las formas de onda obtenidas tras la aplicación de los efectos *Pseudorobot* y *Robotización* tradicional son muy diferentes entre sí. Ambos efectos producen picos cíclicos en la forma de onda pero, mientas que el efecto tradicional deja inalterada la forma de onda entre los picos, el efecto *Pseudorobot* elimina los datos que se encuentran entre los picos.

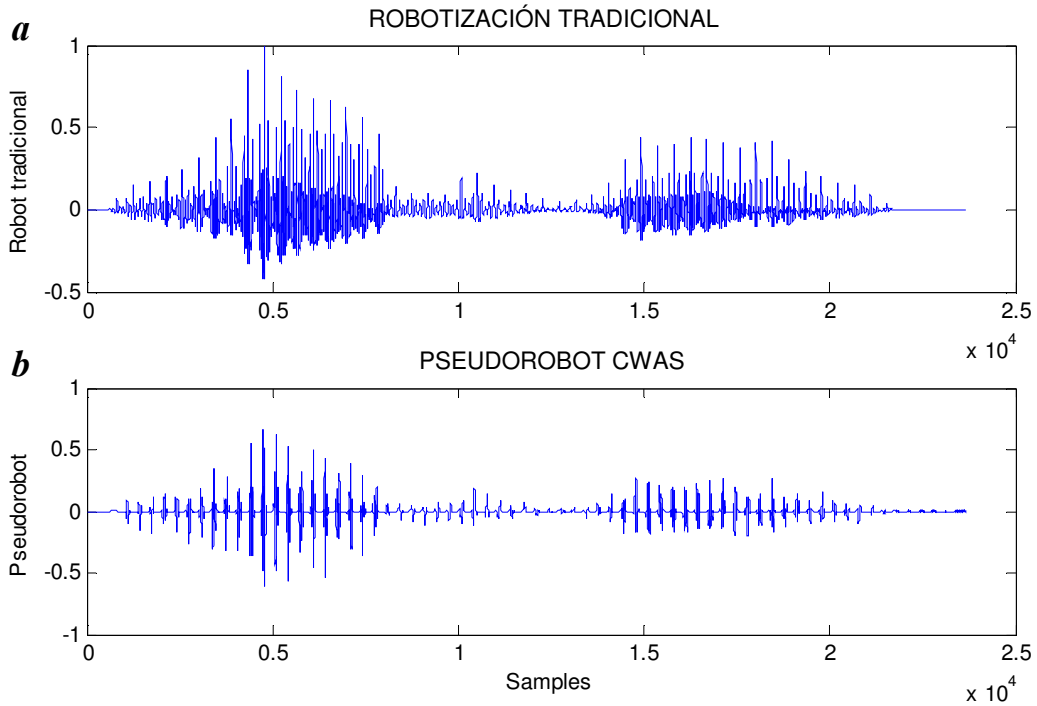


Figura 4.26. Formas de onda. (a) *Robotización* tradicional, (b) Efecto *Pseudorobot* CWAS.

Por otro lado, si se analizan los escalogramas obtenidos para ambos efectos (Figura 4.27) se observa que son muy distintos entre sí. Una de las diferencias más evidentes es el desplazamiento de las frecuencias instantáneas producido con el efecto *Pseudorobot*, igual que ocurría en el efecto de *Robotización* CWAS (apartado 4.5).

Todas estas diferencias hacen que, aunque con ambos métodos se consigan resultados acústicos similares, el efecto *Pseudorobot* no pueda ser considerado como una *Robotización* tradicional.

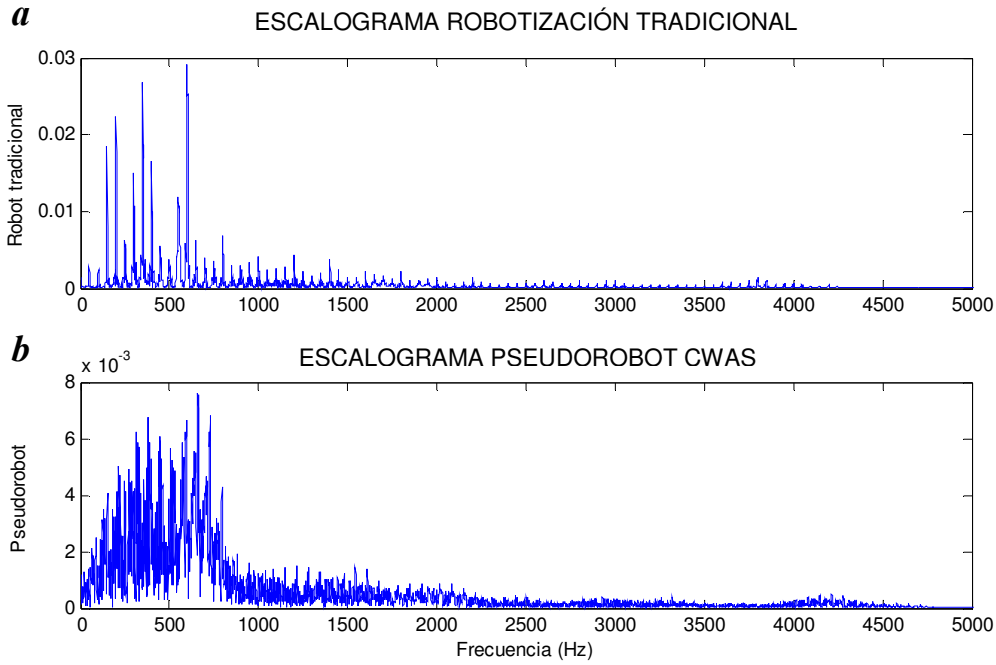


Figura 4.27. Escalogramas. (a) Efecto de *Robotización* tradicional, (b) Efecto *Pseudorobot* CWAS.

4.14. ARMONIZADOR

El efecto *Armonizador* consiste en añadir una quinta justa (7 semitonos) sobre la frecuencia fundamental del sonido. Se recuerda que este efecto se ha implementado por primera vez en este proyecto y se desconoce su implementación mediante métodos tradicionales.

En la Figura 4.28(b) se muestra el escalograma de una nota ejecutada por una trompa tras aplicarle el efecto *Armonizador* CWAS, es decir, tras añadir una nota 7 semitonos por encima de la nota base y atenuada en amplitud respecto a ésta. Los picos correspondientes a la nota añadida se han marcado con puntos rojos.

Como se puede observar en la figura, todos los picos correspondientes a la nota añadida quedan atenuados (porque así se ha querido) y presentan menor energía en el escalograma. Además, puesto que se trata de una nota con una mayor frecuencia fundamental, sus armónicos están más espaciados en frecuencia que los de la nota base.

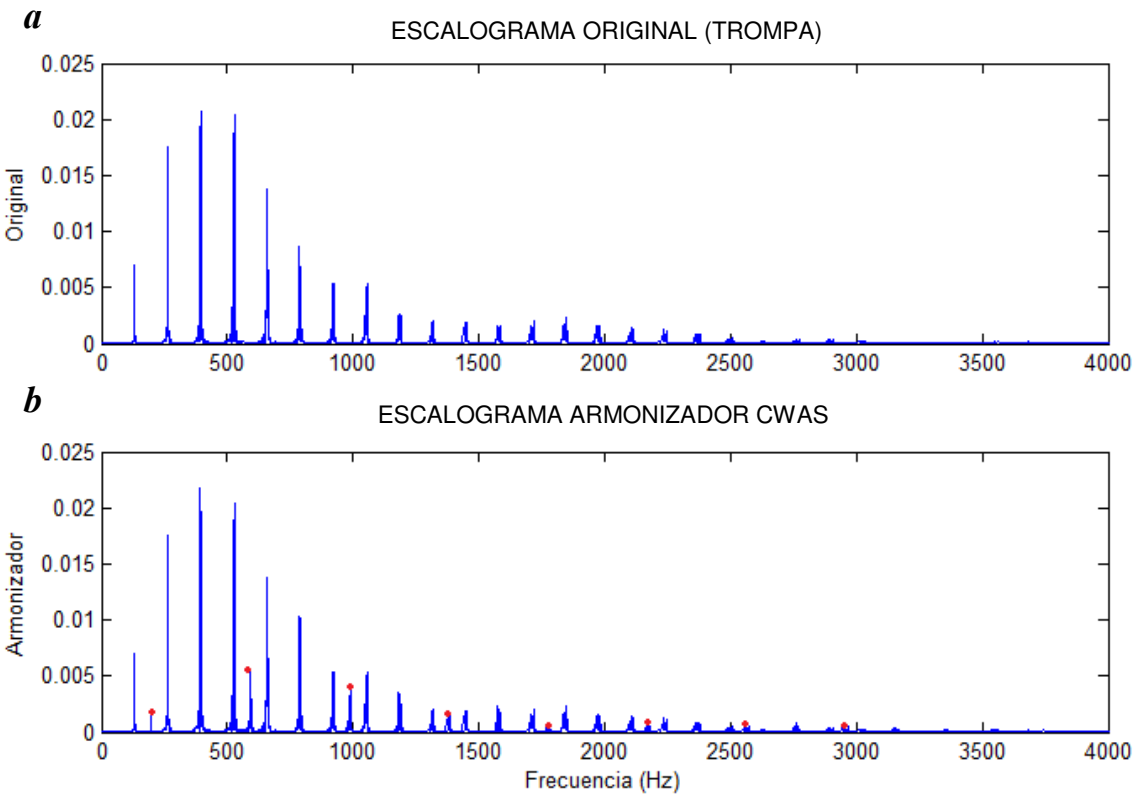


Figura 4.28. Escalogramas del efecto *Armonizador* aplicado sobre el sonido de una trompa. (a) Sonido original, (b) Efecto *Armonizador* CWAS.

Los resultados obtenidos han sido satisfactorios y el algoritmo implementado podría constituir la base de un armonizador inteligente que crease voces más complejas a añadir al sonido original.

4.15. REVERB

A continuación se comparan los resultados obtenidos con los efectos *Reverb* CWAS y *Reverb* tradicional. Para ello se ha elegido un fragmento de la canción del grupo Red Hot Chili Peppers titulada “Otherside”.

Como se puede observar en la Figura 4.29, las formas de onda de los sonidos resultantes de los efectos difieren de la onda original y, a su vez, las formas de onda del efecto tradicional y del efecto CWAS son también ligeramente distintas entre sí.

Las diferencias entre ambos métodos son mínimas y es difícil apreciarlas de forma gráfica. Ambos efectos son muy similares pero, como puede comprobarse en los archivos de audio correspondientes al efecto y que están a disposición del lector, puede destacarse que el efecto CWAS produce una reverberación más definida que el efecto tradicional

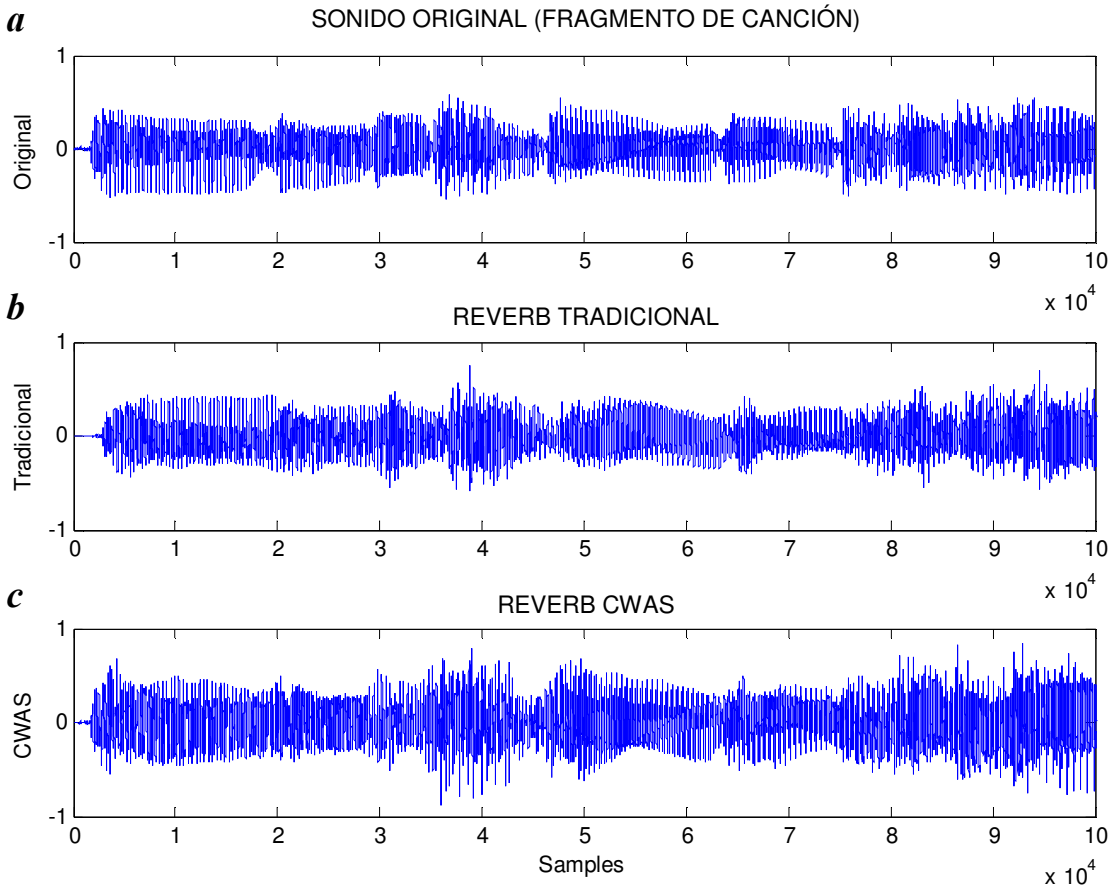


Figura 4.29. Formas de onda del efecto *Reverb*. (a) Sonido original, (b) Efecto *Reverb* tradicional, (c) Efecto *Reverb* CWAS.

4.16. TRÉMOLO

A diferencia de lo que ocurre con otros efectos, la implementación del efecto *Trémolo* a partir de ambos métodos (algoritmo CWAS y método tradicional) ha dado lugar a los mismos resultados. Observando la Figura 4.30 no se aprecia ninguna diferencia significativa entre las formas de onda de los sonidos resultantes de los efectos *Trémolo* CWAS y *Trémolo* tradicional.

Para comprobar que estas diferencias son prácticamente inexistentes, se ha realizado la resta de las dos formas de onda correspondientes a los sonidos resultantes de los efectos. Como se puede observar en la Figura 4.31, las diferencias son mínimas y pueden considerarse despreciables.

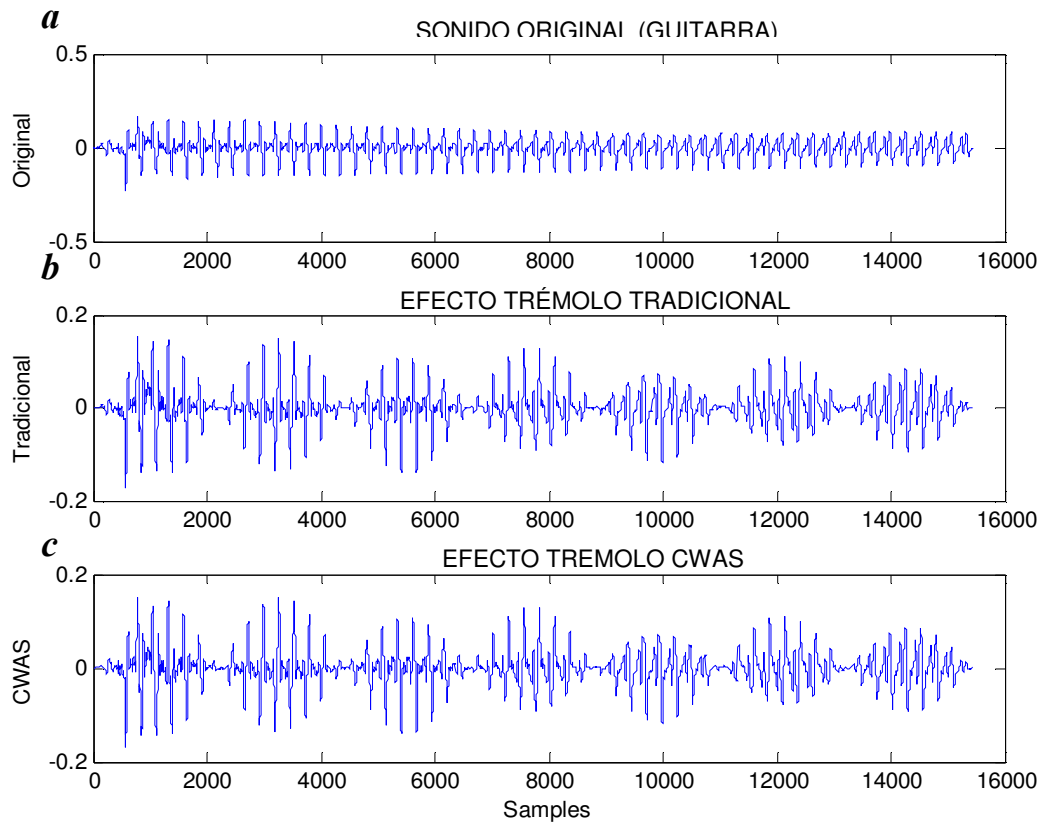


Figura 4.30. Formas de onda del efecto *Trémolo* aplicado sobre el sonido de una guitarra. (a) Sonido original, (b) Efecto *Trémolo* tradicional (c) Efecto *Trémolo* CWAS.

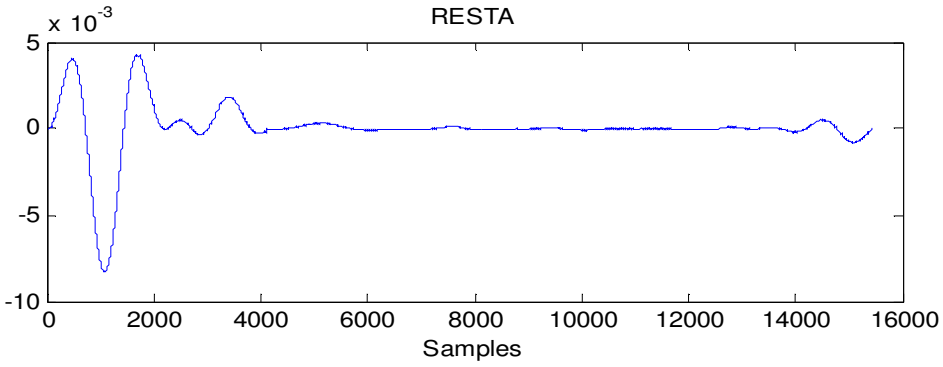


Figura 4.31. Resta de las formas de onda correspondientes a los efectos *Trémolo* tradicional y *Trémolo* CWAS.

De igual manera, si se comparan los escalogramas correspondientes a los efectos *Trémolo* tradicional (Figura 4.32(b)) y *Trémolo* CWAS (Figura 4.32.(c)) puede observarse que también coinciden. En cambio, comparando estos escalogramas con el del sonido original se observan algunas diferencias. Por lo tanto, puede afirmarse que, además de modificar la amplitud del sonido, el efecto *Trémolo* conlleva un ligero cambio en la respuesta frecuencial del sonido original.

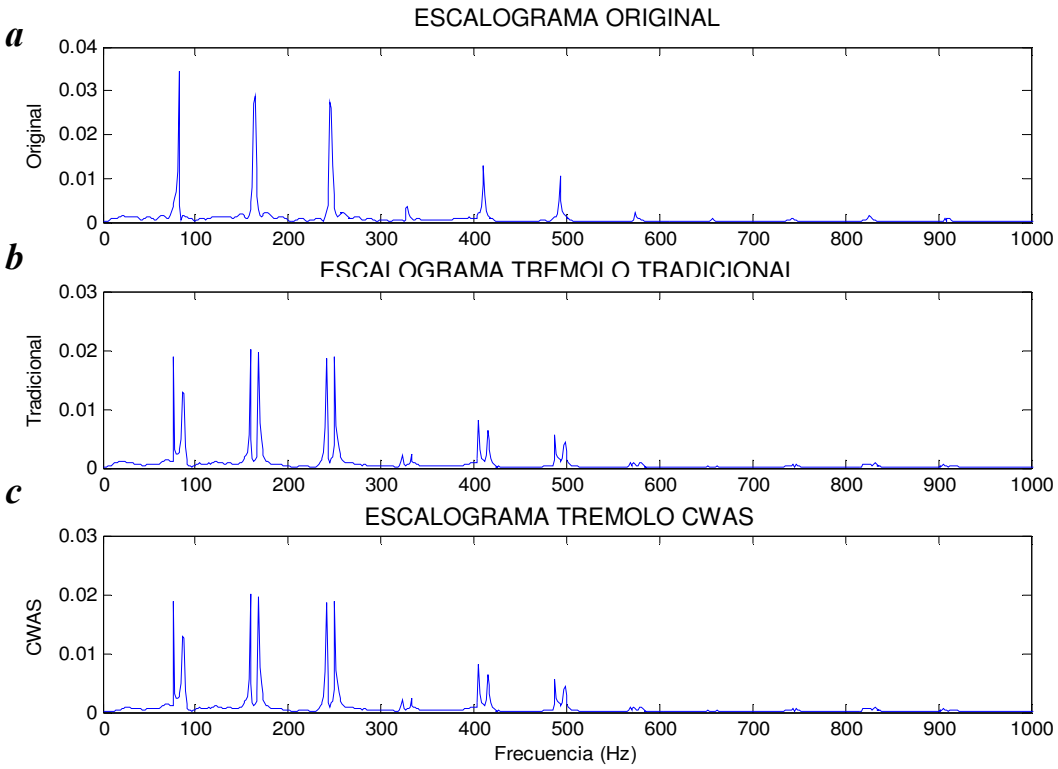


Figura 4.32. Escalogramas del efecto *Trémolo*. (a) Sonido original, (b) Efecto *Trémolo* tradicional, (c) Efecto *Trémolo* CWAS.

4.17. OVERDRIVE Y DISTORSIÓN CON SIMULACIÓN VALVULAR

Para concluir con el apartado de resultados del proyecto se van a analizar los efectos *Overdrive y Distorsión*, teniendo en cuenta que la implementación de dichos efectos con el algoritmo CWAS ha ido acompañada de una simulación de la respuesta con un amplificador de válvulas de vacío.

Overdrive

Comparando las formas de onda de los sonidos (Figura 4.33) se puede observar que, aunque los efectos *Overdrive* tradicional y *Overdrive* con simulación valvular guardan algunas similitudes, existen diferencias ya visibles en la forma de onda.

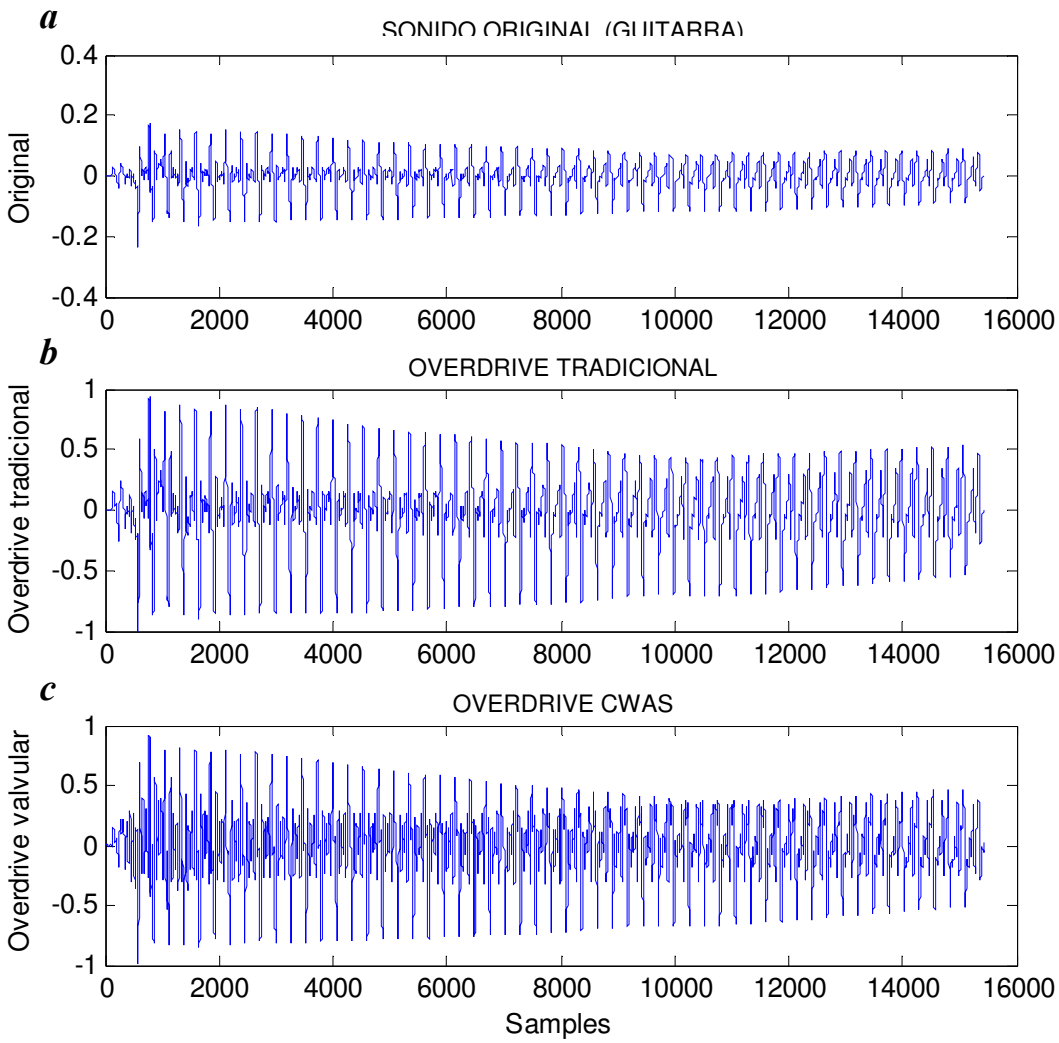


Figura 4.33. Fornas de onda del efecto *Overdrive*. (a) Sonido original, (b) Efecto *Overdrive* tradicional, (c) Efecto *Overdrive* CWAS con simulación valvular.

Para profundizar más en estas diferencias entre las formas de onda hay que analizar también los escalogramas de cada uno de los sonidos.

Como se puede observar en la Figura 4.34(a), en el sonido original están bastante realzados los armónicos 2 y 3 del sonido (son los parciales más importantes en energía junto al primer armónico). Tras la aplicación del efecto *Overdrive* tradicional (Figura 4.34(b)), las proporciones de energía de estos armónicos se mantienen y el escalograma resultante tiene la misma forma que el escalograma original, con la única diferencia de que los picos tienen mayor energía debido al aumento que se produce en la amplitud de la onda. Sin embargo, si se analizan los resultados del escalograma correspondiente al efecto *Overdrive* CWAS (Figura 4.34(c)), se observa un ligero aumento en los armónicos 2 y 3, así como un aumento todavía mayor en el armónico 5, que queda totalmente realzado, mientras que en el sonido original es un armónico con menor presencia energética.

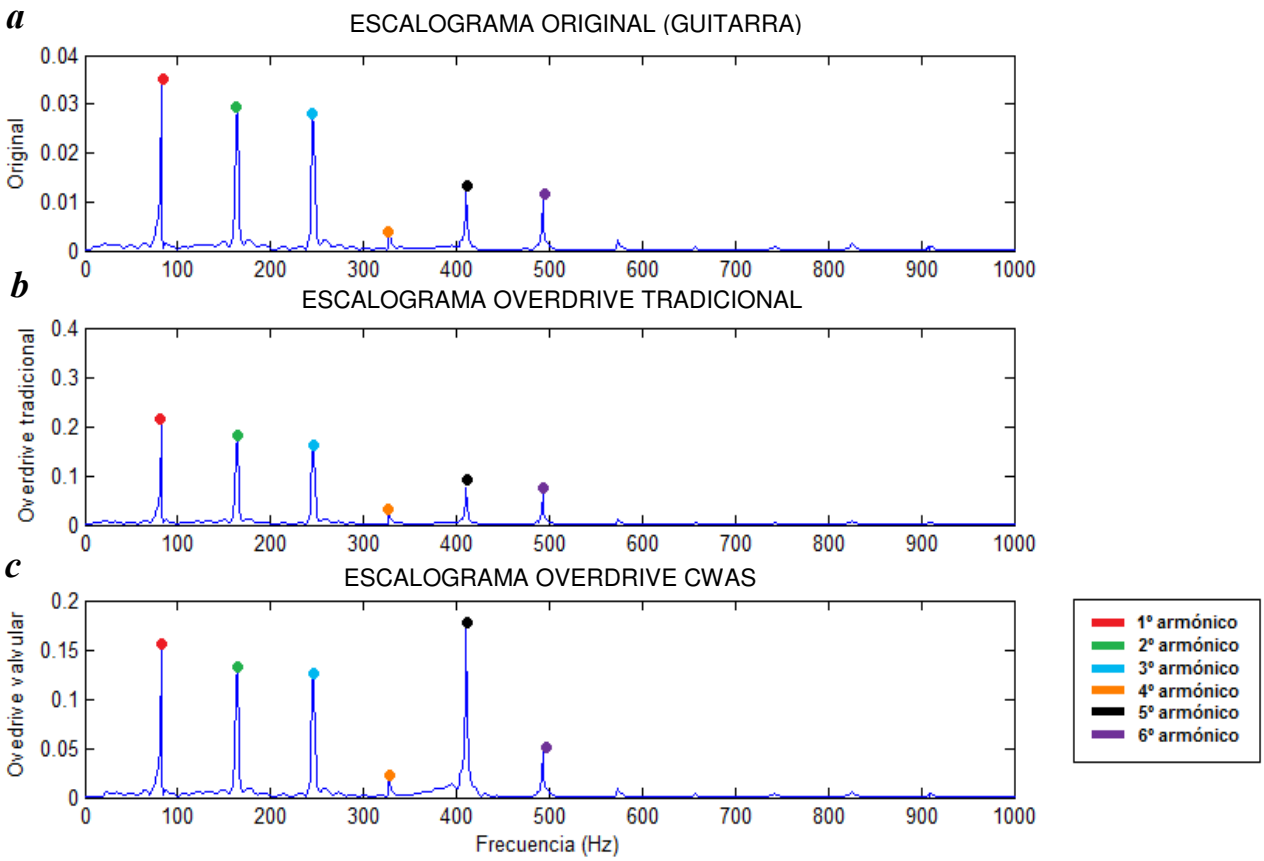


Figura 4.34. Escalogramas del efecto *Overdrive*. (a) Sonido original, (b) Efecto *Overdrive* tradicional, (c) Efecto *Overdrive* CWAS con simulación valvular.

Distorsión

Igual que ocurre con el efecto *Overdrive*, la aplicación del efecto *Distorsión* también produce cambios en la forma de onda del sonido original (Figura 4.35). Sin embargo, comparando las formas de onda correspondientes a los efectos *Distorsión* tradicional y *Distorsión* CWAS, las diferencias entre ambas son mínimas y, a simple vista, el resultado podría parecer el mismo.

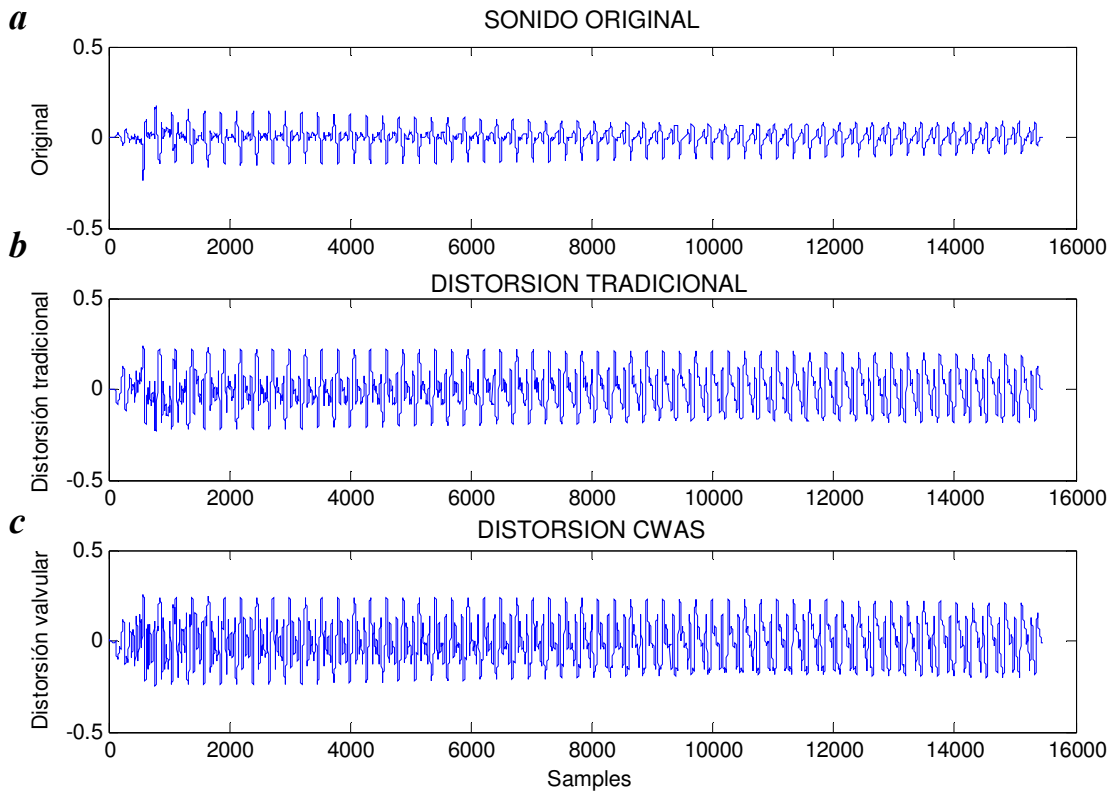


Figura 4.35. Formas de onda del efecto *Distorsión*. (a) Sonido original, (b) Efecto *Distorsión* tradicional, (c) Efecto *Distorsión* CWAS con simulación valvular.

Por otro lado, como se puede observar en la Figura 4.36, el efecto *Distorsión* realza los armónicos 2, 3 y 5 del sonido, siendo este último el más realzado con respecto a sus condiciones originales cuando se aplica el efecto *Distorsión* CWAS.

Otra de las características del efecto *Distorsión* es la adición de armónicos que no existían en el sonido original (de frecuencias superiores a 500 Hz) como consecuencia del importante cambio sufrido por la forma de onda del sonido. En el efecto *Overdrive*, en cambio, este incremento de armónicos no se producía ya que la naturaleza del efecto es más respetuosa con la forma de onda original.

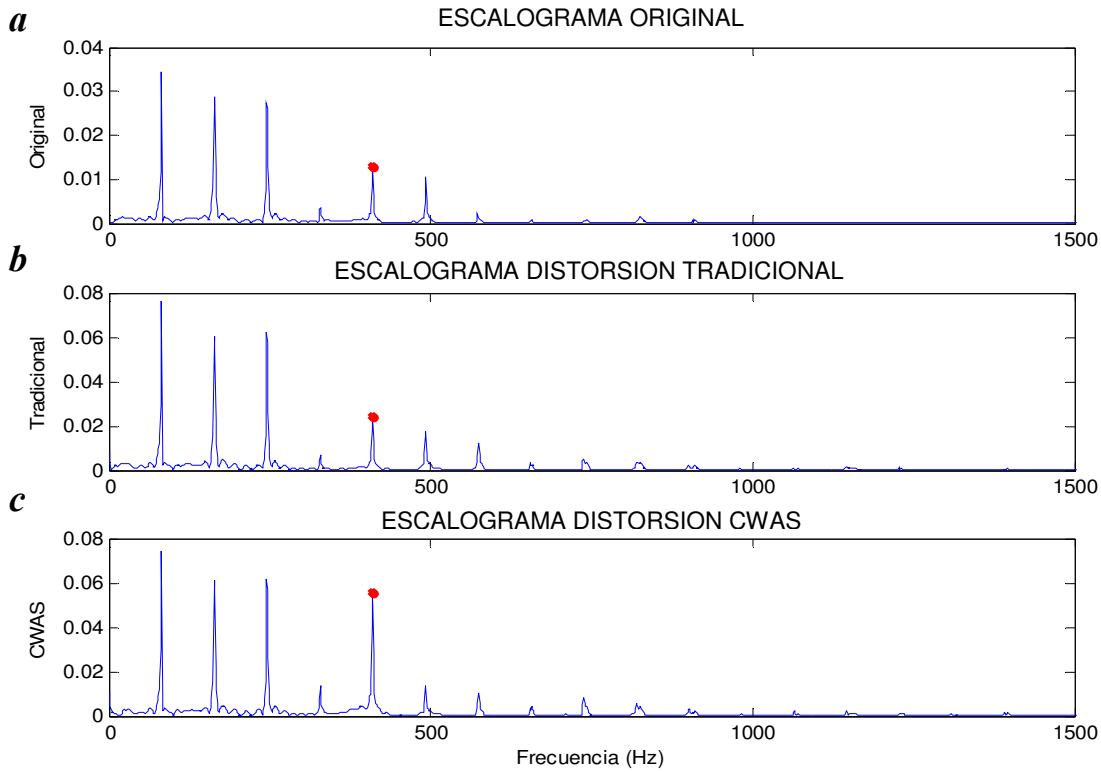


Figura 4.36. Escalogramas del efecto *Distorsión*. (a) Sonido original, (b) Efecto *Distorsión* tradicional, c) Efecto *Distorsión* CWAS con simulación valvular.

Como se acaba de comprobar, la simulación valvular funciona a la perfección en ambos efectos (*Overdrive* y *Distorsión*), realzando los armónicos 2, 3 y 5 del sonido. De esta manera se produce un sonido más agradable y musical.

5. CONCLUSIONES

El presente proyecto fin de carrera ha consistido en la programación alternativa de varios efectos musicales. Para ello se ha utilizado un método de análisis y síntesis de señales de audio desarrollado en la Universidad de Zaragoza, denominado Algoritmo de Síntesis Aditiva por Wavelets Complejas (CWAS por sus siglas en inglés). Con el objetivo de evaluar las posibles mejoras que puede introducir la aplicación del CWAS en la programación de efectos musicales, se ha realizado una comparación de los resultados obtenidos con aquéllos resultantes de la programación tradicional, basados en su mayoría en el uso de la transformada corta de Fourier.

La implementación basada en el algoritmo CWAS de la mayoría de los efectos estudiados ha ofrecido unos resultados acústicos con una calidad mucho mejor que la que se consigue en la implementación tradicional de los mismos efectos.

En ningún caso se ha detectado que el sonido resultante de un efecto implementado por el método CWAS suene peor que su correspondiente efecto tradicional. Sin embargo, de forma excepcional, en alguno de los efectos como el *Vibrato* o el *Trémolo*, se han obtenido resultados muy similares con ambos métodos, sin conseguirse una mejora significativa de la calidad del sonido resultante, de manera que resulta difícil diferenciar entre los sonidos correspondientes a uno y otro método.

La ventaja de utilizar el método CWAS en la implementación de este tipo de efectos sencillos, en los que no se observan grandes diferencias con respecto a la implementación tradicional, es la posibilidad de introducir en ellos ciertas variaciones. Por ejemplo, con la implementación tradicional del efecto *Chorus* no puede conseguirse que, a la vez que se duplica la voz original, cambien las características tonales y frecuenciales de la voz repetida para dar una mayor sensación de realismo. Conseguir esta variación en la implementación tradicional supondría trabajar con un algoritmo mucho más complejo que el algoritmo CWAS y, lo que es peor, el resultado acústico sería peor que el obtenido a partir del método CWAS.

Por tanto, el método CWAS puede suponer una evolución para algunos efectos tradicionales, consiguiendo efectos mucho más complejos y musicales.

También cabe destacar que, bajo el punto de vista del autor del proyecto, otro de los puntos fuertes de este método de análisis del sonido es que permite la clonación de un sonido real con total exactitud, de manera que se puede obtener un sonido sintético indistinguible del original. Esto puede resultar muy interesante para su aplicación, principalmente, en instrumentos como los sintetizadores, que buscan emular el sonido de los instrumentos reales con el mayor grado de acierto posible. En la actualidad estos instrumentos se acercan más o menos al sonido del instrumento al que pretenden imitar, pero es evidente que no suenan exactamente igual.

En definitiva, en un campo tan exigente como el del audio profesional, en el que se busca obtener siempre la mejor calidad sonara posible, el algoritmo CWAS brinda unas posibilidades nunca vistas hasta ahora. Los resultados obtenidos en este proyecto invitan a pensar en la posibilidad de que dentro de unos años el análisis de las señales de audio mediante la CWAS pueda llegar a sustituir a los métodos actuales basados en el uso de las STFT.

6. REFERENCIAS BIBLIOGRÁFICAS

[1]. Javier García de Jalón, José Ignacio Rodríguez, Jesús Vidal, *Aprenda Matlab 7.0 como si estuviera en primero*, Escuela Técnica Superior de Ingenieros Industriales (2005).

[2]. José Ramón Beltrán, Jesús Ponce de León, *Estimation of the instantaneous amplitude and the instantaneous frequency of audio signals using complex wavelets*, Signal Processing 90 (2010) 3093–3109.

[3]. Udo Zölzer, *DAFX- Digital Audio Effects*, Wiley (2002).

[4]. Sergi Jordà Puig, *Audio digital y MIDI*, Guías Monográficas Anaya Multimedia, Madrid (1997).

[5]. Roads, C. *The Computer Music Tutorial*, MIT Press (1996).

[6]. Página web: *www.sonidoyaudio.com*. Última fecha de consulta: 01/08/2011.

[7]. Kevin O'Connor, *The ultimate Tone*, London Power Press (1994).